Studies on the design and construction of an electronic digital computer(Dissertation_全文)

AUTHOR(S):

Yajima, Shuzo

STUDIES

ON

THE DESIGN AND CONSTRUCTION OF

AN ELECTRONIC DIGITAL COMPUTER

SHUZO YAJIMA

STUDIES

ON

THE DESIGN AND CONSTRUCTION OF

AN ELECTRONIC DIGITAL COMPUTER

by

Shuzo Yajima

Kyoto University

July 1963

PREFACE

This thesis reports the results of studies on the design and constructio of an electronic digital computer, achieved particularly through experiences in designing and constructing the Kyoto University Digital Computer KDC-I under the guidance of Professor Ken-ichi Maeda while the author had been a postgraduate student of the Faculty of Engineering of Kyoto University during the year 1958 to 1961. The results are compiled in nine chapters and appendices.

Developments in computer techniques are so rapid that the KDC-I which was at a moment one of the largest computers in this country is considered at present to be a medium scale medium speed computer. Nevertheless, various techniques developed for the KDC-I have been utilized for other computers and measuring instruments, and the KDC-I has been functioning daily since August 1960 at the computation center of the University. The work thus deals with almost all phases of the design and construction, but emphasis is laid on what is beliebed new and interesting. These introductory remarks are found in chapter 1.

Chapter 2 is concerned with the system design of the computer, in which the process of the design of the KDC-I is explained and the outline of the system is described.

Chapter 3 is concerned with the instructions of the computer, in which a list of the KDC-I instructions is given first and details of logical and special operations which are thought new and interesting are described.

Details of the magnetic tape system of the KDC-I are described in chapter 4, in which a new block number system and an instruction system which enables a simple treatment of the drop-out problem of magnetic tape are introduced.

Chapter 5 deals with the problem of the initial read-in routines for the computer, in which the simplest form of reading the routine is suggested along with a proof.

The problem of logical design is discussed in chapter 6. The logical properties of the basic logic circuit of the computer are described first, then the algebraic methods of logical design are reviewed. The simplification of a Boolean function is performed by using the Veitch diagram simplification method for obtaining the simplest normal form. A method devised by the author of obtaining the simplest normal form with a restriction on the operation of an input gate in which no more than a single AND gate can send binary 1 at the same time is described. Next, the main features of the logical design are given, and some arithmetic elements are presented to show examples of the simplest normal function with the gate restriction.

Chapter 7 is concerned with circuits of the computer, in which emphasis is laid on the performance margin of the KDC-I basic logic circuit which uses the dynamic circuitry techniques. A procedure is set up in which the margin of all basic circuits, i.e., thousands of them, not a few samples of them, is measured, and the margin of several functional blocks is then measured The adjustment of the computer became much simpler by applying this procedure together with a thorough examination of the results of logical design and the wiring of the computer. The repetition of a famous ironical sentence "The computer will be completed in less than six months" every six months was no more applied, though it was partly true, if not six months, in writing the thesis.

Measuring the performance margin of thousands of basic logic circuits poses some difficulties. It was conducted point-by-point, which was indeed troublesome and inefficient. Thus the author devised a unit to display the performance margin of the basic logic circuit , the details of which

are given in chapter 8.

In chapter 9 the main points of the results of the studies are presented as a part of the conclusion.

Several items relevant to the subject are added in appendices.

Kyoto — Shuzo Yajima

July 1963

CONTENTS

APPENDIX

Chapter 1

INTRODUCTION

Almost all electronic digital computers designed and constructed today are of the stored program type suggested by the late John von Neumann, and no basically new type computer has as yet been reported. However, the development of the technique of the design and construction of computers is extremely rapid by the efforts of scientists and engineers. A new higher speed and larger scale computer could solve many difficult problems which previously could not be solved. A small-size and light-weight computer has surely an important practical value. It is not too much an exaggeration to say that automation in data handling or information processing has just made a beginning creditable for an infant. There are indeed many things which should be developed much further.

Most theories concerning computing machineries treat least requirements for them, e.g., theories on the Turing machine are typical examples. Computers actually built are much more complex and useful ones in which convenience is an important design factor. Thus the theory of the design for a whole computer has hardly been developed. However, to a tiny well-defined portion of the logic circuit algebraic methods can be applied.

This thesis is concerned with the results of studies on the design and construction of an computer, achieved particularly through experience in designing and constructing the Kyoto University Digital Computer KDC-I (Ku 1,2,3),(Ya 1,2). Developments of computer techniques are so rapid that the KDC-I which was at a moment one of the largest computers in this country is considered at present to be a medium scale medium speed computer. Nevertheless, various techniques developed for the KDC-I have been utilized for other

computers and measuring instruments, and the KDC-I has been functioning daily since August 1960 at the computation center of the University. The work thus deals with almost all phases of the design and construction, but emphasis is laid on what is believed new and interesting.

The KDC-I is a product of a two year project of the University commenced in 1958 aiming to establish a computation center to promote research in the University.

The KDC-I has been developed and built in collaboration with Hitachi, Ltd. and put into operation since August 1960 at the computation center . Magnetic tape units have been added to the computer since October 1960 togethe with a magnetic tape control unit which contains a high speed magnetic core memory.

The programming group at the University has already completed a fairly extensive library of subroutines and very useful assemblers such as SYCS-1 (Ki 1). Moreover a KDC ALGOL 60 Compiler is now under development(Na 1

Several computers had already been constructed at a few research laboratories in this country at the time of the design of the KDC-I. However some of them were reported to be having trouble concerning the reliability of the computers.

One of the design principles was therefore the construction of a very reliable computer, and at the same time a computer which would enable users to program very easily and would be fit for use by many University research workers from various fields.

The actual design of the KDC-I was begun in April 1959. In September the design had taken definite shape. The assembly had been started in Novembe At the end of April 1960, computation had become possible by the main part of the computer. The back panel wiring design of the KDC-I was first done involving many man-hours of work, and then in April it was redone by the

newly operating KDC-I itself and the results were compared with its built-in panel wiring (Ya 9).

The process of adjustment was believed promising in which, among ninety-five varieties of one-and-a-half address instructions, a record has been established of adjusting thirty instructions in a day and various floating-point instructions have been put into operation with almost no failures in logics and in wiring in spite of its being a new model and a first product, which we believe has indicated progress in computer desig n and construction technique.

The adjustment of the magnetic tape units and the magnetic tape control unit consumed almost three months because of lack of experience until various magnetic tape test programs produced by the Kyoto University Programming Group (Ku 4) were operated correctly.

The KDC-I has already performed a vast amount of calculations on vaious problems which have been submitted by the staff and students of the University.

With two years of operating experience, we are gaining satisfactory results from this computer.

In the subsequent three chapters the system, the instruction system and the magnetic tape system of the computer are described. The problem of the initial read-in routine is discussed in chapter 5. In chapter 6 the logical design of the computer is described. The circuits of the computer and the problem of the performance margin are presented in chapter 7, and a unit to display the performance margin of the basic logic circuit of the computer is treated in chapter 8. Concluding remarks are given in chapter 9.

Chapter 2

PROPOSED ORGANIZATION OF THE COMPUTER

2.1 The Design of the Kyoto University Digital Computer KDC-I

Basic factors of the system design of a computer are (1) system requirements, (2) characteristics of feasible computer components, (3) the techniques known, e.g., the methods of logical design, etc., (4) the economics of the situation and (5) policy restrictions, like most other engineering problems. In the case of designing a special-purpose computer, system requirements are a very clear factor. However, in the case of designing a general-purpose computer, the weight of the latter factors described above increases.

In the case of the KDC-I, the second factor was serious, since some of a not many computers being constructed in this country were reported to be having trouble concerning their reliability. The main electronic circuits of these computers used vacuum tubes, parametrons and transistors. As could be foreseen, solid-state circuits were showing more favorable performance. Among them transistor circuits have been considered most promising with regard to their reliability, their smaller power dissipation, their flexibility in regard to circuit design, their speed of operation, and so on. Thus transistor circuits have been chosen for the main circuits of the KDC-I.

The most important design principle was therefore the construction of a very reliable computer, and at the same time a computer which would enable users to program very easily and would be fit for use by many University research workers from various fields. The computation speed of the computer was not necessarily considered of prime importance at the time of designing since above all a reliable computer had to be developed.

The important items of the computer system had been roughly decided

by the Kyoto University and the manufacturer at the time when the author joined in the project. Main items of them were;- (1) that it must be a decimal computer, (2) that it performs internal floating-point operations, (3) that the instruction system must be the one-and-a-half address system, (4) that the length of the word is 12 digits including sign, (5) the use of the 6,000 rpm magnetic drum as a 4,200 word main memory of the computer (6) the use of the diode logic and the transistor dynamic circuit which operates at the clock rate of about 230 kc/s as a basic logic circuit, (7) the use of the photo-electric tape reader as an input device, (8) the use of the typewriter as an output device, (9) the use of the magnetic tape as an external memory, and so on.

Most of the items were regarded appropriate in view of the technical level of the computer at that time. The chief reason for a decimal system was that the manufacturer wanted a general-purpose computer, though the author thought a binary system was more advantageous in case of the scientific use at the University. The use of the typewriter as an output device was one of the most serious problems because of its low operation speed. To solve this problem, a high speed punching device was planned to be installed some time in the future.

One more serious problem to the plan was the relation between the one-and-a-half address instruction system and the use of the 6,000 rpm magnetic drum as a main memory of the computer, because of the unbalance between the operation time of the instructions and the memory access time. The operation time of most of the instructions were estimated at the order of several hundred micro-seconds while the average memory access time was 5ms. The delay line type quick access memory whose average access time was 1.25 ms was added, and the situation had considerably been improved. It was expected that the high speed random access core memory would replace the drum as a main memory of this computer in the future. A portion of

this expectation was fulfilled by using the 50-word core matrix memory to the magnetic tape control unit of the computer. It was very difficult to construct a large scale core matrix memory by using domestic memory cores at the time of the design, and the memory cores for the 50 word memory were imported from the General Ceramic Co. of the United States of America.

The most anxious problem was the design of the magnetic tape memory because of lack of experience.

Concerning the computer instructions, it was also very roughly suggested, but it was almost made anew and several new and interesting instructions were added. Details are given in the chapter 3.

The main items of the computer system were thus decided and the work for the detaild design had then been started. The main design principles of the system are as follows;-

(1) The building-block scheme was adopted, in which each unit, the central processing unit, the magnetic tape control unit, the magnetic tape units, etc., constituted a building-block. A high speed memory unit and high speed output equipment should be connected to the computer in the future.

(2) To ensure the reliability of the computer, parity checking and validity checking (a redundancy checking of the BCD code) were adopted. Even parity is examined at every storage register and an input register, whereas validity errors are checked at all registers. The odd character parity and the even channel parity are examined for each information block on the magnetic tape.

(3) To increase the overall processing speed of the computer, the operation of the output and of the magnetic tape units is made concurrent with that of the central processing unit.

(4) The magnetic tape units are connected to the central processing unit via a magnetic tape control unit in which a magnetic core memory is contained and used as a buffer storage as well as a high speed memory to the central

processing unit.

(5) The programming ought to be as simple as possible for the general users in the University.

(6) The speed of the operation of the floating-poing and shifting instructions is made high in comparison with that of other instructions.

(7) At the operator's console, the operating condition of the computer is supervised and various controls are also possible.

(8) The adjustment and maintenance of the computer must be simple. For example a transistor dynamic flip-flop circuit together with diode logic circuits is mounted on a plug-in type printed card.

(9) The same code is used for the magnetic tape as for the paper tape, thus keeping both codes consistent.

2.2 The Outline of the Computer

Before describing further details of the computer, the outline of the computer is presented in this section.

The details of the operation and use of the computer are described in references (Ku 1,2,3). A view of the KDC-I is shown in Fig.2.1.

(1) Type of the computer: A stored program electronic digital computer.

(2) Type of information accepted by the computer: Alphanumerical characters encoded in 8 unit excess-16 binary number on perforated paper tape at the input and output (I/O) of the computer. In the computer, a series of 12 decimal digits forms a word, and each decimal number is binary coded and treated in parallel.

(3) Word: A word represents an instruction, a fixed or floating point number or a character word. In Fig. 2.2 the structure of a word is illustrated.

(4) Instruction system: One and a half address. 95 varieties of instructions including 9 instructions for magnetic tapes. The function to clear the contents of the Accumulator can be added to all instructions, if necessary.

(5) Operation registers and counters: A 23-digit double length Accumulator (AC, consisting of an Upper Accumulator UA and a Lower Accumulator LA), a Multiplicand-Divisor register (MD), and Order Register (OR), three 4-digit Index Registers (IR 1, 2, 3) and a 4-digit Instruction Location Counter (LC), which are all accessible from programmers. Besides, there are a temporary Multiplier-Quotient Register (MQ), an Output Buffer Register (BR), an Input Register (IPR) and an Output Register (OPR). Fig. 2.3 shows these registers and counters.

(6) Memory: 4,250 addressable storage registers in total, plus magnetic tape memory.

i) Magnetic core memory: 50 words with 50 us access time, which also

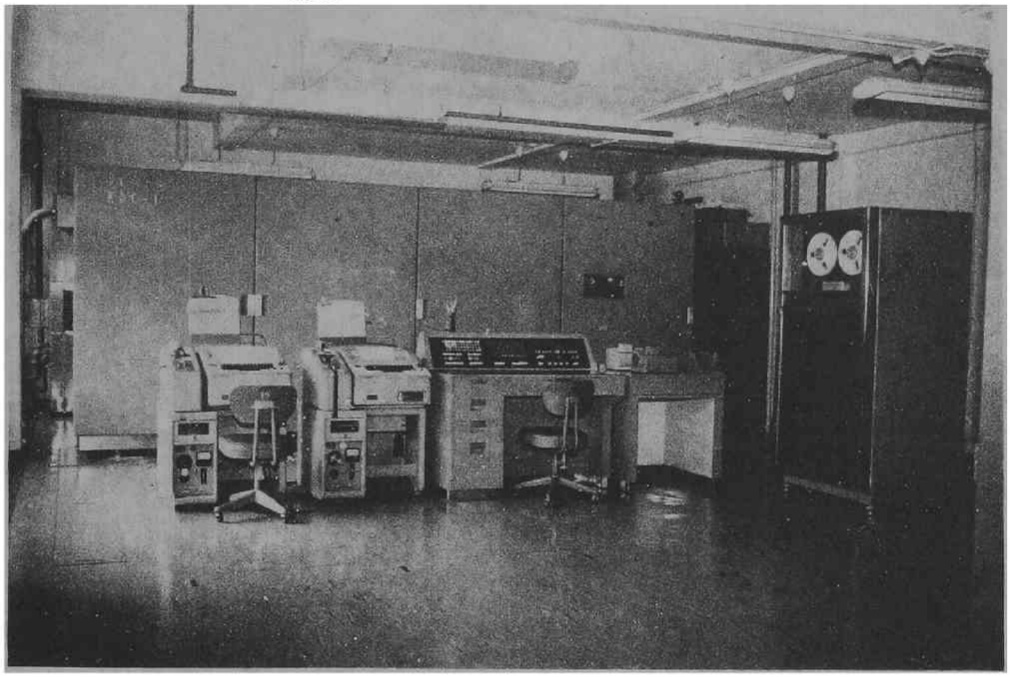


Fig.2.1-View of the KDC-I.
In foreground from left to right: the off-line typewriter, the console typewriter, the operator's console, the photo-electric tape readers and the magnetic tape handler.
In background from left to right: the central processing unit and the magnetic tape control unit.

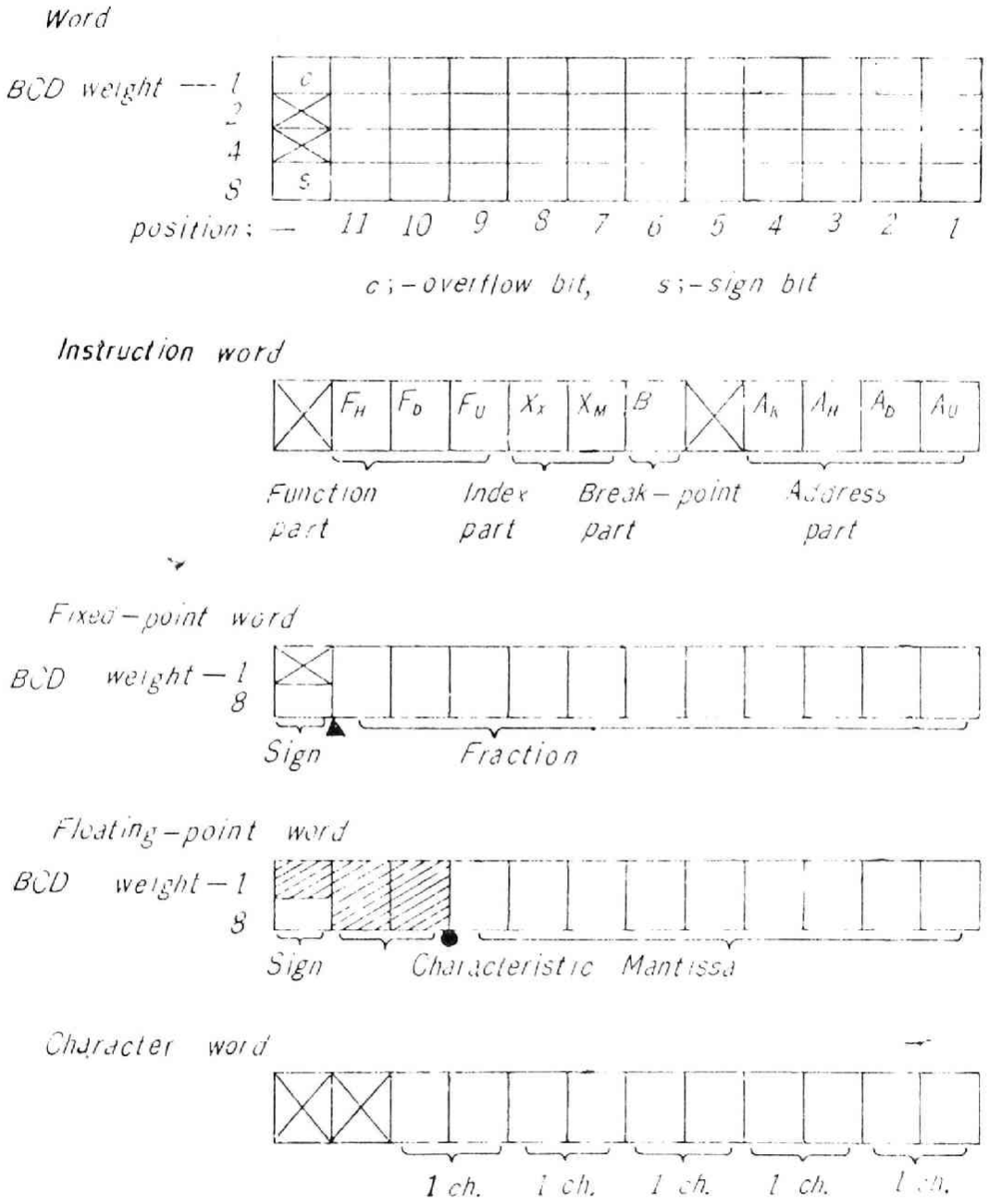
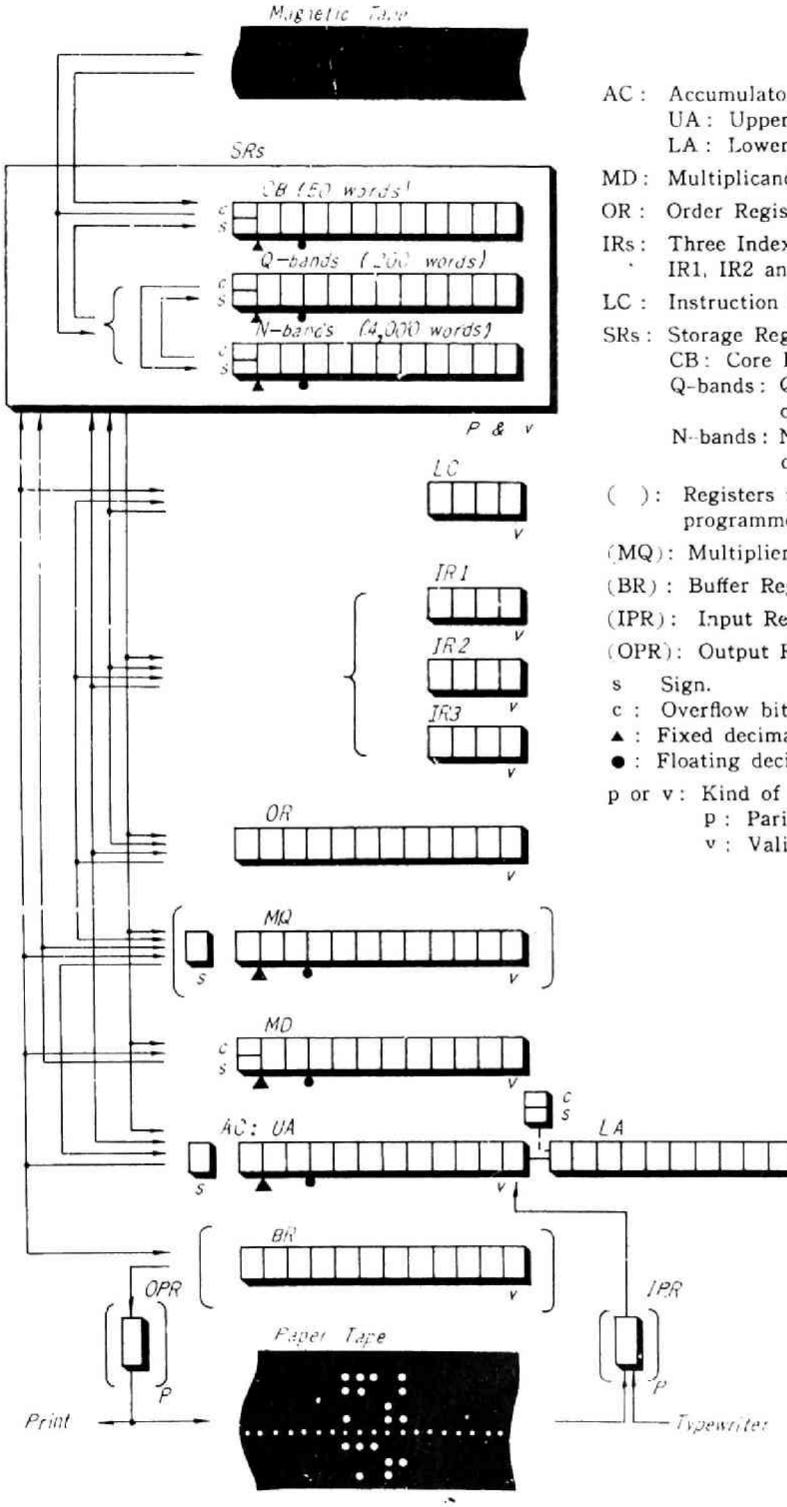

Word
BCD weight — 1
2
4
8
c
s
position: — 11 10 9 8 7 6 5 4 3 2 1
c ; – overflow bit, s ; – sign bit
Instruction word
F_H F_D F_U X_x X_M B A_h A_H A_D A_U
Function part
Index part
Break – point part
Address part
Fixed – point word
BCD weight – 1
8
Sign
Fraction
Floating – point word
BCD weight – 1
8
Sign
Characteristic
Mantissa
Character word
1 ch. 1 ch. 1 ch. 1 ch. 1 ch.

Fig.2.2-Word.



AC : Accumulator,
UA : Upper Accumulator,
LA : Lower Accumulator.

MD : Multiplicand-Divisor Register.

OR : Order Register.

IRs : Three Index Registers :
IR1, IR2 and IR3.

LC : Instruction Location Counter.

SRs : Storage Registers,
CB : Core Buffer Memory,
Q-bands : Quick access bands of the drum,
N-bands : Normal access bands of the drum.

( ) : Registers not accessible from programmers.

(MQ) : Multiplier-Quotient Register.

(BR) : Buffer Register.

(IPR) : Input Register.

(OPR) : Output Register.

s Sign.
c : Overflow bit.
▲ : Fixed decimal point.
● : Floating decimal point.

p or v : Kind of checking,
p : Parity check,
v : Validity check.

Fig.2.3 Main registers and counters and main information paths.

serves as a buffer storage for the magnetic tape.

ii) Magnetic drum memory: 4,000 words on 6,000 rpm magnetic drum with the average access time of 5 ms. 200 words in delay-line type quick access memory with 1.25 ms average access time on the same drum.

iii) Magnetic tape memory: As many as four magnetic tape units can be connected to the computer, two units have been attached to the computer at present. A standard reel of magnetic tape contains about 1,100 m(3,600 ft) Mylar base magnetic coating tape, capable of storing about 7,000 50-word block, or 350,000 words, with the block-transfer-time from 220 ms minimum to 12 minites maximum.

(7) Circuit: Diode logic and trasistor dynamic flip-flop circuits using 3 kinds of clock pulses and operating in a single phase at 230 kc per second (Ni 1),(Ya 7).

(8) Operation time: (excluding only memory access, the time for the modification of an address is included)

Addition: 0.5 ms for double length fixed point.
1.3 ms for double length floating point.

Multiplication: 5.8 ms average for fixed point.
5.2 ms average for floating point.

Division: 6.0 ms average for fixed point.
5.1 ms average for floating point.

(9) On-line input-output: Two photo-electric tape readers: 200 characters per second each.

Console typewriter: 8 characters per second.

High speed punch is scheduled to be installed in near future.

(10) Magnetic tape: The same code as for paper tape. 9,600 characters per second. 6.4 bits per mm. Off-line processors, such as magnetic tape to high speed printer, are not available at the University

at present.

(11) Organization: The computer consists of the followings (refer to Fig.2.4);-

1. Central processing unit (containing magnetic drum memory)
2. Operator's console
3. Photo-electric tape readers .... 2 sets
4. Console typewriters .... 1 set
5. Off-line typewriters .... many

   High speed punch .... (1)* set

5. Magnetic tape control unit (containing magnetic core memory)
6. Magnetic tape units .... 2 (4)* sets

* The number of sets scheduled for installation.

The extention of magnetic core memory and high speed printer is our next objectives to the computer.

Power is supplied to the computer through moter-generator voltage regulators.

The temperature of the machine room is kept at around 20 degree C.

Further information necessary in the subsequent chapters is described in the following sections.

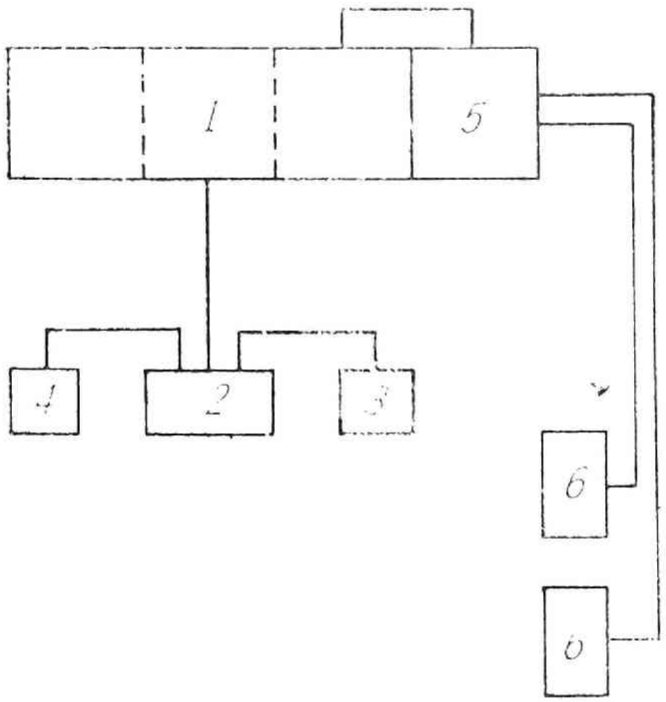


Fig. 2.4-Organization of the KDC-I.

1: Central processing unit.
2: Operator's console.
3: Photo-electric tape readers.
4: Console typewriter.
5: Magnetic tape control unit.
6: Magnetic tape handlers.

2.3 Forms of Information Accepted by the Computer

1. Input/Output (I/O) Characters

At the input or the output of the computer, alphanumerical characters are used like an english-letter typewriter. These characters are coded in 8 unit (excess-16) binary number.

The input device reads these codes which are expressed on a proper perforated paper tape. Or these characters are read directly from the keys of the on-line typewriter of the computer.

The output device prints and/or punches (perforates a proper paper tape) these characters.

These KDC-I I/O characters and the paper tape character coding are listed in Table I and II of Appendix A respectively.

However, in the computer, only decimal numbers with signs are used. So a conversion takes place when information are read in and written out. Further details are described in the section "Input-Output Operations and Tape Control Codes" of the next chapter.

2. Binary Coded Decimal (BCD)

The KDC-I is a decimal computer. Every decimal number in the computer is expressed in a binary coded decimal form (BCD), in which four bits (binary digits) constitute a decimal number and each bit has its weight 1, 2, 4 or 8.

Details are given below;-

decimal number		0	1	2	3	4	5	6	7	8	9
BCD	weight 1	0	1	0	1	0	1	0	1	0	1
	weight 2	0	0	1	1	0	0	1	1	0	0
	weight 4	0	0	0	0	1	1	1	1	0	0
	weight 8	0	0	0	0	0	0	0	0	1	1

The + or - sign is expressed by 0 or 1 of a binary number respectively.

A word "set" is used to place a number 1 (or -) in a bit, while a word

"reset" is used to place a number 0 (or +) in a bit.

3. Words

In the computer a series of 12 decimal digits is treated as a unit in the memory or it forms a word. Each decimal digit is expressed in the BCD and treated in parallel.

A word thus represents actually a 11-digit number with an overflow bit (the weight 1 of the BCD number) and a sign (the weight 8 of the BCD number). Two bits in the weight 2 and 4 of the BCD number are not used at all.

Ordinary registers of the computer have the capacity to store a word. There are 4,250 storage registers and some operation registers in the computer.

An instruction, or a fixed or a floating point number takes the form of a word. So each is often called an instruction word or a data word respectively. A data word is usually expressed in a fixed or a floating number.

Typical usage of a word is shown in Fig. 2.2.

Negative number is not expressed in a complement form, but is expressed with the magnetude and a minus sign.

4. Fixed-Point Numbers

A fixed-point number as a data word is a 11-digit decimal fraction with a + or - sign. Beside, a word has one more extra bit i.e. an overflow bit, but it is neglected in most of the fixed point operations.

Thus a fixed point number represents a 11-digit number in the range which is greater than -1 and smaller than -1. Two kinds of zero, +0 and -0, exist, but in most cases +0 is used.

If the overflow bit is carefully used, it is not impossible to expand the range from (+1, -1) to (+2, -2) with a great difficulty.
Ordinary registers have a capacity of a word, but one of the operation registers Accumulator (AC) has 23 digits with the AC sign (an overflow digit + 11×2 digits) or a double length of a word.

The multiple length is performed by programming.

5. Floating-Point Numbers

A floating-point number as a data word is a 9-digit decimal fraction or mantissa with a 2-digit and 1-bit (an overflow bit) decimal characteristic and a + or - sign (which is shown in Fig. 2-Organization of a Word).
So if a mantissa with a sign is m, and an exponent is t, then a number x is;-

$x = m * 10^t$; $(0.1 \leqq) \ |m| < 1 \quad -100 \leqq t \leqq +99$

There are two signs i.e. a sign of a number and a sign of an exponent. To escape from the double sign, a characteristic (C) which is an exponent plus 100 is used;-

$c = t + 100 \qquad 0 \leqq c \leqq 199$

which needs 2 digits and 1 bit. This bit can be regarded in a sense an exponent sign.

In the floating point operations, a floating point number is treated in a normalized form i.e. in which the mantissa has no heading zeros. If the mantissa is all zeros, a number is regarded zero, but its normalized form is equal to the fixed point form zero, or a word has all zeros, or $0 * 10^{-100}$.

Thus a floating point number x in a normalized form represents a number in the range; $0.1 \times 10^{-100} \leqq |x| < 0.1 \times 10^{100}$ and $|x| = 0$.

Since the Accumulator (AC) has 23 digits, a double length mantissa is treated in the operation, and an exponent overflow or underflow can be treated to some extent.

6. Instruction Words

There are 95 varieties of instructions which can be decoded and executed by the computer. The operation of these instructions are described in the "Computer Instructions". Only the form of an instruction is described here. An instruction is coded in a number and occupies a word. So an instruction word and a data word are same in apperarence. But locations in which these words are stored are easily traced. The computer traces and executes

instructions in sequence specified by the contents of the Instruction Location Counter (LC) as is described in "The Operation of the Computer".

An instruction has a 3-digit function part (F_H, F_D, F_U), a 2-digit index part (X_X, X_M), a 1-digit break point part (B), and a 4-digit address part (A_K, A_H, A_D, A_U), as is shown in Fig. 2-Organization of Words. Other digits of an instruction word are neglected in the operation of the instruction.

The function part decides the type of operation performed and the address part gives the necessary information in the operation. A number in the address part can be modified by the contents of the Index Registers (IR) specified by the index part of the instruction. In this sense the instruction system of the computer is called one and a half address system.

2.4 Registers and Counters

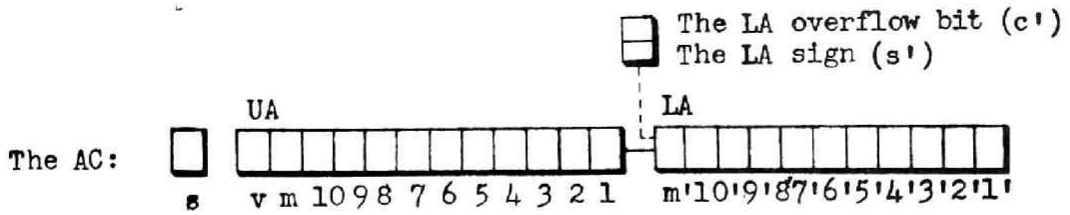
Informations are processed at and/or stored in registers. An register has ordinarily a capacity of a word. Among operation registers and counters, there are AC, MD, OR, IRs and LC. The main memory has 4,250 addressable Storage Registers (SRs). All will be described in turn.

All these registers and counters are accessible from the programmers i.e. the contents can be read and any necessary number of a proper size can be placed in them by programming. Beside, there are various registers and counters which help the operation of instructions, but they are not accessible from programmers. Among these are a temporary Multiplicand-Quotient Register (MQ, 12 digits and a sign) which is used for various purposes, an Output Buffer Register (BR, 12 digits), an Input Register (IPR) and an Output Register (OPR). These registers and counters behind the scene are not described here.

1. Accumulator (AC), Upper Accumulator (UA) and Lower Accumulator (LA)

The AC plays a most important role in the operation of the computer. The AC is used for various purposes. For example, all results by arithmetic operations are placed in the AC. The Upper Accumulator is used for the storage for the input and output operations and so on.

The AC has 23 digits (an overflow digit and 11 × 2 digits) and a + or - sign (AC sign), while an ordinary storage register has 11 digits, an overflow bit and a sign. Since the upper half (12 digits) and lower half (11 digits) of the AC are often used separately, each half has its own name, i.e. the Upper Accumulator (UA) and the Lower Accumulator (LA) respectively. Accordingly the UA has 12 digits (an overflow digit and 11 digits) and the AC sign, while the LA has 11 digits. Since the LA is used for the storage of the remainder as a result of the division operation, a LA sign and a LA overflow bit exist only for this purpose outside the LA part of the AC. The name of each digit of the AC is as follows;-



s : the AC sign (+ or -; reset to 0 or set to 1 respectively)

v : the overflow digit

m : the most significant digit of the UA

1 : the least significant digit of the UA

m': the most significant digit of the LA

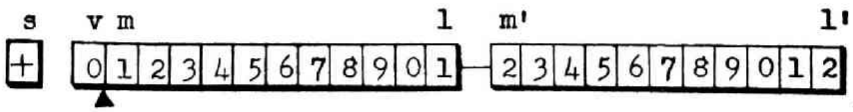
1': the least significant digit of the LA

s': the LA sign (+ or -; 0 or 1 respectively)

c': the LA overflow bit (the first digit of the characteristic of a floating-point number in the LA)

As is described in the Fixed Point Numbers, the range of a fixed point number is (-1, +1). Since the AC has an overflow digit, its range is (-10, +10). The state of an overflow can be detected by various ways, and it will be described in the Computer Instructions. The decimal point in case of a fixed point number is therefor between the v and m positions of the AC.

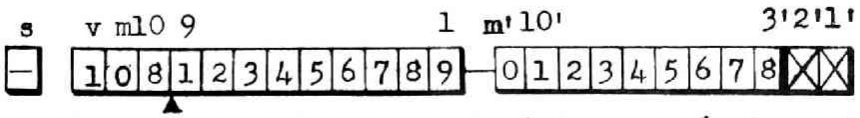
e.g.



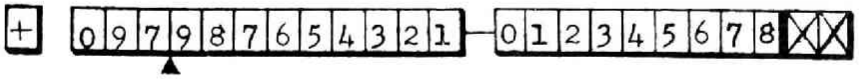
c(AC) = + 0.123 456 789 012 345 678 901 2

In case of a floating point number, the decimal point comes between the 10 and 9 position of the AC. The length of the mantissa is 18 digits from the 9 position to the 3' position. Numbers in the 2' and 1' positions are neglected or reset to zero in most of the floating point operations. The characteristic (v, m, 10) has 3 digits, but if it becomes 200 or larger than 200, an exponent overflow is said to have occured, and is detected by various ways, and it will be described in the Computer Instructions.

e.g.



$c(AC) = -\ 0.123\ 456\ 789\ 012\ 345\ 678 \times 10^{8}$ (c=108)



$c(AC) = +\ 0.987\ 654\ 321\ 012\ 345\ 678 \times 10^{-3}$ (c=97)

When the LA is used separately, the decimal point comes between the 1 and m' position of the AC in case of a fixed point number, or between the 10' and 9' position of the LA in case of a floating point number. Details are found in the description of the instructions DVJ and FDJ.

The 4, 3, 2 and 1 positions of the UA are also called the address part of the UA.

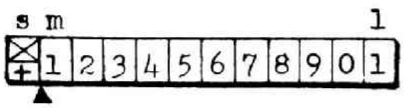
2. Multiplicand-Divisor Register (MD)

The MD is used for the storage of a multiplicand or a divisor, and for other various purposes such as for comparison, an extractor and a temporary storage.

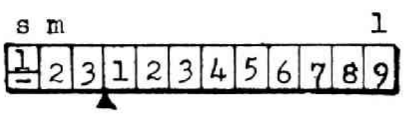
The MD has a capacity of a word i.e. 11 digits, an overflow bit and a sign bit, quite the same form of a storage register.

The name of each digit and each weight is therefore the same as that for a word.

e.g.



c(MD) = fx(+ 0.123 456 789 01)



$c(MD) = fl(-\ 0.123\ 456\ 789 \times 10^{23})$

3. Order Register (OR)

The computer reads an instruction word and places it in this Order Register OR, and executes it by decoding each part of the instruction. The OR has a capacity of a word. The name of the positions of digits of the OR is therefore the same as that of the instruction word in Fig. 2- Organization of a Word.

It is possible to set manually an instruction in the OR and to execute it at the console of the computer. This function is used for various ways such as the insertion of an instruction in the program, setting the proper number in the location counter LC and so on.

4. Index Registers (IRs; IR1, IR2, IR3)

The Index Registers (IRs) are mainly used for the modification of the address of an instruction, counting of the repetition of some operations and so on.

There are three index registers; IR1, IR2 and IR3. Each has a capacity to store a 4-digit decimal number without signs. Counting up and down is made in modulo 10,000 fashion. All IRs can be used equally, but in the operation of the instruction JXU, the comparison of the contents of the IRs are made only with the IR1.

5. Instruction Location Counter (LC)

As is described in the Operation of the Computer, the Instruction Location Counter (LC) is used to control the sequence of programs. An instruction word in the location specified by the contents of the LC is read and executed. Then one is added automatically to the contents of the LC in the normal sequence, so the next instruction is read and executed and so on. The control instructions change the contents of the LC conditionally or unconditionally and the computer takes the next instruction from there and proceed from there (or the control jumps).

The LC has 4 digits without sign. If the contents of LC contains a

number other than 0000~4249, zeros replace the whole contents of the OR and the computation stops.

The contents of the LC is also used for the modification of the address of the instruction whose location is specified by the contents of the LC. This time, the LC is treated just like the 4th index register or IR4 (to specify the LC, 4 is used, while the IR1, 2 and 3 are specified by 1, 2 and 3).

6. Storage Registers (SRs)

There are 4,250 Storage Registers in total as a main memory, each of which has its own address (0000-4,249). A main memory is augmented by an auxiliary memory or a magnetic tape memory. This item is described in the Magnetic Tape Memory. Magnetic cores and a magnetic drum constitute a main memory. Each storage register has a capacity of a word i.e. 12 digits. An extra parity bit is added to each decimal digit (even parity) when information is stored (a word: 5 bits parallel 12 digits series). The parity check is made when reading these informations. The access time of each storage register depends on the type of the memory.

(1) The magnetic drum memory

Information is magnetically stored on the surface of the rotating drum (ca. 6000rpm). Each five out of many Read/Write heads write or read words on or from the magnetic drum. It needs ca. 10ms per one revolution of the drum. The periphery of the magnetic drum under these five heads stores 200 words (2,400 bits per head), and is called a normal band of the magnetic drum (N-band). There are 20 normal bands (or 4,000 storage registers). Therefor the average access time is ca. 5ms.

Beside these, there are four quick access bands (Q-band). One quarter of the periphery of the magnetic drum is used for a Q-band, and has the capacity of 50 words with the agerage access time of ca. 1.25ms (total 200 words). In each Q-band five write heads write words, and five read heads spaced a quarter of the peripheral length apart from the write heads in the

direction of the revolution read these words. Then five erase heads erase (or reset all number to zeros) all these words. Information read is fed back to the write heads, so the information recirculates in each Q-band.

To provide an address to each location of these storage registers, a Drum Location Counter (DL) counts numbers in synchronism with the revolution of the drum. Since 200 words exist in the periphery, the DL counts from 0 to 199 in modulo 200 fashion per revolution, and for the Q-band the contents of the DL is decoded in modulo 50 fashion.

To access to the storage register of a certain address, the computer selects a certain band and waits till the DL counts a certain number.

The address (A_K, A_H, A_D, A_U) of each storage register is as followes;

Normal Bands: (0000 - 3999) (average access 5ms)

N-band 1 : 0000 - 0199

2 : 0200 - 0399

... ...

20 : 3800 - 3999

Quick Access Bands (4000 - 4199) (average access 1.25ms)

Q-band 1 : 4000 - 4049

2 : 4050 - 4099

3 : 4100 - 4149

4 : 4150 - 4199

In the N-band, A_U, A_D and the weight-1 of the A_H (or A_H, A_D, A_U in modulo 200) is used for the coincidence with the contents of the DL. Other information is used for the selection of the N-bands.

In the Q-band, A_D and A_U in modulo 50 is used for the coincidence with the contents of the DL in modulo 50. Other information is used for the selection of the Q-bands.

(2) The magnetic core memory

Ir has a capacity of 50 words and is a random-access memory with 50μs

access time. Information are stored in the magnetic cores arranged in matrix in one bit per one core basis. Just the same way as of the drum memory, parity bit is added to each digit of a word. The parity of information are checked whenever information are read.

The address is; 4200 - 4249.

The usage of the core memory is suspended during the most of the magnetic tape operations. Detainls are described in the Magnetic Tape Operations.

2.5 Error Checks

Two kinds of checks are employed in the computer; a parity check and a validity check.

(1) The parity check at the Input

A code of each I/O character has a (even) parity bit, which makes the number of 1s in the code even. If an odd number of 1s in the code is detected, the computation stops at the end of the operation of the current instruction, and sets on the HALT lamp and the INP PARITY CHECK lamp on the console.

(2) The parity check at the Storage Registers

Each decimal code is coded in the BCD form with four bits. An extra bit or a even parity bit is added to this four-bit code. When reading information from the SRs, a parity is examined. If an odd number of 1s in the code is detected, the computation stops at the end of the operation of the current instruction, and sets on the HALT lamp and the DM or CORE PARITY CHECK lamp on the console depending upon the type of memories in which the error has originated.

N.B. In Q-band of the drum memory, the parity is constantly examined.

(3) The validity check at all the registers and counters

In the BCD, 10 out of 16 states of a 4-bit number are used, and other 6 states are therefor a forbidden combination of a 4-bit number. This forbidden combination is constantly check by the computer, and this check is called a validity check. This check is made in all the operation registers and counters and the storage registers. When this kind of errors is checked, the computation stops at the end of the operation of the current instruction, and sets on the HALT lamp and the VALIDITY CHECK lamp on the console.

(4) The dual parity checks of the magnetic tape memory.

This item is described in the Magnetic Tape Memory.

(5) These check lamps are extinguished by depressing the START or SS button

on the console which at the same time starts the operation of the next instruction in sequence. But if a parity error is being checked at the Q-bands of the drum memory, or a validity error is being checked at the operation registers, and since these checks are constantly made, the computation stops once again by these checks unless these error should be removed.

Chapter 3

INSTRUCTIONS OF THE COMPUTER

3.1 Introduction

The most suitable set of instructions to the computer used by the staff and students of Kyoto University cannot be decided uniquely. At first Hitachi, Ltd. suggested the one which was based on a set of instructions developed for the ETL Mark V computer of Electrotechnical Laboratory of Japan. It became the basis of discussion between Kyoto University and Hitachi, Ltd. at the beginning. However, it was pointed out that there were many ambignities in definitions of the instructions, and moreover there were too many arithmetic operations in comparison with non-arithmetic operations such as shifting, transfer, control, index and logical operations.

The original plan was composed of about 130 instructions, and after a series of joint discussions it became almost a new set of instructions.

Professor T. Sakai of Kyoto University emphasized the importance of logical operations in connection with the problem of pattern recognitions. The author analysed the suggestions and reached to a conclusion that not only the operation between words or between decimal digits but also the operation between numbers of different weights and between bits are necessary in spite of the decimal structure of the machine. The instructions which perform logical operations were thus developed. The author who had to perform logical design of the computer and for that purpose was sent to Tozuka Works of Hitachi, Ltd. from the University had made a final draft of the KDC-I instructions for Hitachi, Ltd., and the draft was then suggested by Hitachi, Ltd. to Kyoto University and it was finally approved by both with some corrections.

The instruction system of the computer is one-and-a-half address or a single address with the address modification function by the contents of

the IRs. Ninity-five varieties of instructions including nine instructions for magnetic tapes was finally adopted. The function to clear the contents of the AC can be added to all instructions, if necessary.

By providing a rather rich repertory of instructions and a decimal structure to the machine, the programming for the computer is expected to become much easier than for existing machines.

In this chapter main features of the instructions, and a list of KMC-I instructions are presented in the first place, and then the details of logical and special instructions, which are thought new and interesting, will be described. The details of the definition of all KMC-I instructions will be given in Appendix B.

3.2 A List of KDC-I Instructions

Before presenting a list of instructions, necessary information is given for the convenience of describing the computer instruction as concisely as possible.

1. Symbolic Codes

(1) The symbolic representation of an instruction

As is described in the instruction word, the necessary information in case of describing its operation are a function part (F_H, F_D, F_U), an index part (X_X, X_M), and an address part (A_P, A_H, A_D, A_U). A break point has rather an independent function, so it is enough to describe it once separately.

Every instruction has its own 3-alphabet mnemonic symbolic code, e.g. the "subtract" operation has its symbolic code SUB, and its numerical code is 10_4 or the function part of this instruction must be 10_4. The address part are described by a symbolic address A, and if the modification of the address is possible by the IR1 and J, the instruction is written as SUB 1J A, etc. (PAU .. -; in the operation of this instruction, the numbers in the address and the index parts are neglected in the operation.)

(2) Symbols concerning the index part of an instruction

The index part (X_X, S_M) of an instruction is used for various ways. But mainly this part is used to specify the index registers in question. The IR1 is specified by 1, IR2 by 2, and IR3 by 3, or symbolically IRJ is specified by J, and so on. Since this part is used for various purposes, the following symbols are often used according to the usage of the part, though the usage of any symbols is allowed;-

I,J : In case of the modification of an address.

N.B. I or J can be 1, 2, 3 or 4 (c(LC) is used). But in some situation I, J, K and L are used for IR1, 2, 3 and LC respectively

H : In case of the Index Operations where the IR on which the operation is made should be specified by the Xx part (H = 1, 2, 3).

M : In case of the instruction TLU where the location to which the control might jump should be prepared in the c(IR M) (M = 1, 2, 3).

Q : In case of the instructions LDQ, STQ, where the number of a quick band should be specified by the Xx part (Q = 1, 2, 3, 4; irrelevant to the IRs).

S : In case of the instruction JSW where one out of five JSW switches on the console should be specified by the Xx part (S = 1, 2, 3, 4, 5; irrelevant to the IRs).

N : In case of the Magnetic Tape Operations where one out of four magnetic tape handlers should be specified (N = 1, 2 at present; irrelevant to the IRs).

(3) Symbols concerning the address part of an instruction

A : a symbolic address (actually four digits)

E : an effective address or modified address (actually four digits)

n : a symbolic number which specifys a number of shift, a number of characters, etc. This n is written in the address part of an instruction, but it does not specify an address.

m : a type of a character and a number of multitude in the instruction MSP.

2. The Modification of an Address

A number in the address part of an instruction can be modified by the contents of the IR1,2,3 and/or the LC specified by each number in the two-digit index part of an instruction. The index part of an instruction is used for various ways as described in the previous paragraph.

In the description of each computer instruction, the symbol I and/or J are used when the index part is used to specify the IRs or LC for the modification, and the symbol H is used when the index part is used to specify the one of the IRs on which the operation is made. The modified address is often called the effective address, denoted as E.

The modification is made just before the execution of the current

instruction at the OR. The c(IR) or c(LC) specified by the symbol J is added to the address part of the instruction in modulo 10,000 and then the c(IR) or c(LC) specified by the symbol I is added to the address part of the instruction in modulo 10,000.-

If the modification is unnecessary, zero should be written in each of the X_M or A_X position. If the symbol I or J =1,2,3 or 4, the IR1,2,3 or the LC is specified respectively.

e.g.1.	ADD IJ A	E = #IJA ≡ c(I) + c(J) + A
	ADD 12 A	E ≡ c(IR1) + c(IR2) + A
	ADD 34 A	E ≡ c(IR3) + c(LC) + A
	ADD 22 A	E ≡ 2*c(IR2) + A
	ADD 01 A	E ≡ c(IR1) + A
	ADD 00 A	E = A
	SLS IJ n	#IJn ≡ c(I) + c(J) + n
e.g.2.	JXL HJ A	E = #JA ≡ c(J) + A
	JXL H2 A	E ≡ c(IR2) + A
	JXL H4 A	E ≡ c(LC) + A
	JXL H0 A	E = A
	SEX HJ n	#Jn

3. The Break Point

The Break Point part of an instruction is used when the computation of a program should be stopped at the pre-determined instruction for convenience of a program check.

The computation is stopped at the end of the operation of the instruction whose Break Point part has an odd number, only when the B button on the console is set on, and the HALT lamp on the console is turned on. If the START button on the console is depressed, the next instruction in sequence is started.

Thus if the B Button is not set, the computation is not stopped regardless of a number in the Break Point part.

4. KDC-I Instructions

In Table 1 a list of the KDC-I instructions is presented. The numerical operation code, the symbolic code, the operation time and the explanation of each instruction are described in the ascending order of the numerical operation codes. The details of the definition of all KDC-I instructions will be given in Appendix B.

TABLE 1. KDC-I INSTRUCTIONS

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
100	**ADD I J A**	0.50 + Ai + Aa	Add. c(AC)pUA + c(≠IJA) → c(AC).
102	**ADA I J A**	0.50 + Ai + Aa	Add Absolute. c(AC)pUA + |c(≠IJA)| → c(AC).
104	**SUB I J A**	0.50 + Ai + Aa	Subtract. c(AC)pUA − c(≠IJA) → c(AC).
106	**SBA I J A**	0.50 + Ai + Aa	Subtract Absolute. c(AC)pUA − |c(≠IJA)| → c(AC).
110	**ADM · ·**	0.50 + Ai	Add MD. c(AC)pUA + c(MD) → c(AC).
114	**SBM · ·**	0.50 + Ai	Subtract MD. c(AC)pUA − c(MD) → c(AC).
120	**MPA I J A**	5.8 av + Ai + Aa	Multiply and Add. c(AC) + c(MD)*c(≠IJA) → c(AC).
122	**MPS I J A**	5.8 av + Ai + Aa	Multiply and Subtract. c(AC) − c(MD)*c(≠IJA) → c(AC).
130	**RAA I J n**	0.50 + Ai	Raise Address. c(AC)pUAad + ≠IJn → c(AC).
134	**LWA I J n**	0.50 + Ai	Lower Address. c(AC)pUAad − ≠IJn → c(AC).
138	**RND I J n**	0.50 + Ai	Round. c(AC)rnd → c(AC).
140	**ADR I J A**	6.9 av + Ai + Aa	Add, Divide and Round or Halt. [{c(AC)pUA + c(≠IJA)}/c(MD)]rnd → c(UA); 0 → c(LA), or Add and Halt.
150	**DVJ I J A**	6.0 av + Ai	Divide or Jump. c(AC)/c(MD) → c(UA); Rem → c(LA), or Jump, [R].
152	**DRJ I J A**	6.6 av + Ai	Divide and Round or Jump. {c(AC)/c(MD)}rnd → c(UA); 0 → c(LA), or Jump.
160	**ADL I J A**	0.55 + Ai + Aa	Add to LA. c(AC)pLA + c(≠IJA) → c(AC).
162	**AAL I J A**	0.55 + Ai + Aa	Add Absolute to LA. c(AC)pLA + |c(≠IJA)| → c(AC).
164	**SBL I J A**	0.55 + Ai + Aa	Subtract from LA. c(AC)pLA − c(≠IJA) → c(AC).
166	**SAL I J A**	0.55 + Ai + Aa	Subtract Absolute from LA. c(AC)pLA − |c(≠IJA)| → c(AC).
200	**FAD I J A**	1.3 + Ai + Aa	Floating Add. c(AC) + c(≠IJA) → c(AC).
202	**FAA I J A**	1.3 + Ai + Aa	Floating Add Absolute. c(AC) + |c(≠IJA)| → c(AC).
204	**FSB I J A**	1.3 + Ai + Aa	Floating Subtract. c(AC) − c(≠IJA) → c(AC).
206	**FSA I J A**	1.3 + Ai + Aa	Floating Subtract Absolute. c(AC) − |c(≠IJA)| → c(AC).
210	**FAM · ·**	1.3 + Ai	Floating Add MD. c(AC) + c(MD) → c(AC).
214	**FSM · ·**	1.3 + Ai	Floating Subtract MD. c(AC) − c(MD) → c(AC).
221	**FMP/ I J A**	5.2 av + Ai + Aa	Clear and Floating Multiply. c(MD)*c(≠IJA) → c(AC).

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
223	**FMC/ I J A**	5.2 av + Ai + Aa	Clear, Floating Multiply and Change Sign. $-c(MD)*c(\#IJA) \rightarrow c(AC)$.
238	**FRD I J n**	0.55 + Ai	Floating Round. $c(AC)rnd \rightarrow c(AC)$.
240	**FAV I J A**	6.7 av + Ai + Aa	Floating Add, Divide and Round or Halt. $[\{c(AC)+c(\#IJA)\}/c(MD)]rnd \rightarrow c(UA)$; $0 \rightarrow c(LA)$, or Add and Halt.
250	**FDJ I J A**	5.1 av + Ai	Floating Divide or Jump. $c(AC)/c(MD) \rightarrow c(UA)$; Rem $\rightarrow c(LA)$, or Jump, [R].
252	**FDR I J A**	5.6 av + Ai	Floating Divide and Round or Jump. $\{c(AC)/c(MD)\}rnd \rightarrow c(UA)$; $0 \rightarrow c(LA)$ or Jump.
300	**STO I J A**	0.35 + Ai + Aa	Store. $c(UA) \rightarrow c(\#IJA)$.
302	**SLA I J A**	0.35 + Ai + Aa	Store LA. $c(LA) \rightarrow c(\#IJA)$, N.B. R-ind.
304	**STM I J A**	0.35 + Ai + Aa	Store MD. $c(MD) \rightarrow c(\#IJA)$.
306	**STA I J A**	0.35 + Ai + Aa	Store Address. $c(UA)ad \rightarrow c(\#IJA)ad$.
308	**STL I J A**	0.35 + Ai + Aa	Store Location. $c(LC) \rightarrow c(\#IJA)ad$.
310	**PAM · ·**	0.35 + Ai	Place UA to MD. $c(UA) \rightarrow c(MD)$.
320	**LDM I J A**	0.35 + Ai + Aa	Load MD. $c(\#IJA) \rightarrow c(MD)$.
322	**LDA I J A**	0.35 + Ai + Aa	Load Address. $c(\#IJA)ad \rightarrow c(UA)ad$.
324	**CMP I J A**	0.55 + Ai + Aa	Compare. $c(LC)+1, 2, 3 \rightarrow c(LC)$, if $c(MD) \gtreqless c(\#IJA)$.
340	**FSL I J A**	0.95av + Ai + Aa	Floating Store LA. $c(LA) \rightarrow c(\#IJA)$.
360	**TLU MJ A**	5.5 av + Ai + Aa	Table Look Up. $loc(c(MD)) \rightarrow c(OR)ad$, [P], or $c(M) \rightarrow c(LC)$.
364	**LDQ QJ A**	2.9 + Ai + Aa	Load Quick Access. $c(\#JA), \cdots \rightarrow c(Q\#JA), \cdots$
366	**STQ QJ A**	2.9 + Ai + Aa	Store Quick Access. $c(Q\#JA), \cdots \rightarrow c(\#JA), \cdots$
410	**FFL · · –**	0.80 + Ai	Fixed to Floating. fx(c(AC)) to fl(c(AC)).
450	**FFX I J A**	0.55 + Ai	Floating to Fixed or Jump. fl(c(AC)) to fx(c(AC)), or Jump.
500	**EAD I J A**	0.55 + Ai + Aa	Extract and Add. $c(AC)pUA + c(\#IJA)$ & $c(MD)odd \rightarrow c(AC)$.
502	**ERE I J A**	0.35 + Ai + Aa	Extract and Replace. $c(UA)$ & $c(MD)even$ U $c(\#IJA)$ & $c(MD)odd \rightarrow c(UA)$.
510	**SSP · ·**	0.30 + Ai	Set Sign Plus. $\lvert c(AC) \rvert \rightarrow c(AC)$.
512	**CHS · ·**	0.30 + Ai	Change Sign. $-c(AC) \rightarrow c(AC)$.

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
514	**NOP** · · –	0.30 + Ai	No Operation.
516	**NOT** · · –	0.35 + Ai	NOT. $\overline{c(UA)}$ & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
518	**SCT** · · –	1.4 av + Ai	Shift and Count. LLS until 1st non-zero appears in UAm, and shift count → c(OR)ad, [P].
520	**AND** I J A	0.35 + Ai + Aa	AND. {c(UA) & c(#IJA)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
522	**IOR** I J A	0.35 + Ai + Aa	Inclusive OR. {c(UA) U c(#IJA)} & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).
530	**SLS** I J n	0.55 + Ai	Short Left Shift. left shift of c(UA) by $(\#IJn)_{2,1}$ places.
532	**LLS** I J n	0.55 + Ai	Long Left Shift. left shift of c(AC) by $(\#IJn)_{2,1}$ places.
534	**LCS** I J n	0.55 + Ai	Long Cyclic Shift. cyclic left shift of c(AC) by $c(\#IJn)_{2,1}$ places.
536	**SRS** I J n	0.55 + Ai	Short Right Shift. right shift of c(UA) by $(\#IJn)_{2,1}$ places.
538	**LRS** I J n	0.55 + Ai	Long Right Shift. right shift of c(AC) by $(\#IJn)_{2,1}$ places.
550	**WAN** I J n	0.35 + Ai	Weighted AND. c(UA, 1) & c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).
552	**WOR** I J n	0.35 + Ai	Weighted OR. c(UA, 1) U c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).
630	**SEL** I J n	0.35 + Ai	Select Component. select I/O Component specified by #IJn.
632	**RIN** I J n	PTR; 200 ch/s	Read-in. read $(\#IJn)_{2,1}$ char. into c(UA) in mode specified by $(\#IJn)_4$.
634	**WRT** I J n	0.35 + Ai; 8 ch/s	Write. print and/or punch $(\#IJn)_{2,1}$ char. from c(UA) in mode specified by $(\#IJn)_4$.
636	**WSP** I J m	0.35 + Ai; 8 ch/s	Write Special. print and/or punch a char. specified by $(\#IJm)_{4,3}$ $(\#IJm)_{2,1}$ times.
638	**FWR** · · –	0.35 + Ai; 8 ch/s	Floating Write. print and/or punch fl(c(UA)).
710	**HJM** I J A	0.35 + Ai	Halt and Jump. halt and #IJA → c(LC).
712	**JSW** SJ A	0.35 + Ai	Jump by Switch. #JA → c(LC), if Switch-S is on.
714	**JMP** I J A	0.35 + Ai	Jump. #IJA → c(LC).
750	**JMI** I J A	0.35 + Ai	Jump on Minus. #IJA → c(LC), if c(AC) < −0.
752	**JUN** I J A	0.35 + Ai	Jump on UA No Zero. #IJA → c(LC), if c(UA) ≠ 0.
754	**JNZ** I J A	0.35 + Ai	Jump on No Zero. #IJA → c(LC), if c(AC) ≠ 0.

TABLE 1. (Continued)

NUM. CODE	INSTRUCTION	TIME (ms)	TITLE and OPERATION
756	**JOV IJ A**	0.35 + Ai	Jump on Overflow. #IJA → c(LC), if c(UA)v ≠ 0.
758	**JEO IJ A**	0.35 + Ai	Jump on Exponent Overflow. #IJA → c(LC), if c(UA)v ≧ 2.
820	**LXA HJ A**	0.35 + Ai + Aa	Load Index from Address. c(#JA)ad → c(H).
822	**STX HJ A**	0.35 + Ai + Aa	Store Index. c(H) → c(#JA)ad.
830	**SEX HJ n**	0.35 + Ai	Set Index. #Jn → c(H).
832	**RAX HJ n**	0.35 + Ai	Raise Index. c(H) + #Jn → c(H).
834	**LWX HJ n**	0.35 + Ai	Lower Index. c(H) − #Jn → c(H).
850	**JXL HJ A**	0.35 + Ai	Jump with Index Lowered. if c(H) ≠ 0, #JA → c(LC), and c(H) − 1 → c(H), or if c(H) = 0, normal seq., and 9,999 → c(H).
852	**JXR HJ A**	0.35 + Ai	Jump with Index Raised. if c(H) ≠ 0, #JA → c(LC), and c(H) + 1 → c(H), or if c(H) = 0, normal seq., and 1 → c(H).
854	**JXU HJ A**	0.35 + Ai	Jump with Index Unequal. if c(H) ≠ c(IR1), #JA → c(LC), and c(H) + 1 → c(H), or if c(H) = c(IR1), normal seq., and c(H) + 1 → c(H).
856	**JSX HJ A**	0.35 + Ai	Jump after Set Index from LC. c(LC) → c(H), and #JA → c(LC).
860	**PSX IJ A**	0.35 + Ai + Aa	Set Pseudo Index. c(#IJA)ad → c(OR)ad, [P].
910	**BTP NJ A**	0.45 + Ai ; 220	Buffer to Tape. c(CB) and #JA → c(MT, N ; #JA)
912	**TPB NJ A**	17.0 + Ai ; 220	Tape to Buffer. c(MT, N) → c(CB) ; if blook No. = #JA, skip next instruction, [TC].
914	**BLS NJ A**	0.45 + Ai ; 100*n + 120	Blook Search. c(MT, N ; #JA) → c(CB), [TC].
920	**DMB ·J A**	2.90 + Ai + Aa	Drum to Buffer. c(#IJA), ··· → c(4,200), ···
922	**BDM ·J A**	2.90 + Ai + Aa	Buffer to Drum. c(4,200), ··· → c(#IJA), ···
930	**RWD N· –**	0.45 + Ai ; 20 ; av. ca. 450 cm/s	Rewind. rewind (MT, N) to its load point.
932	**BST N· –**	0.45 + Ai ; 220	Back Space Tape. test c(MT, N) in backward direction, [TC].
934	**TTP N· –**	0.45 + Ai ; 220	Test Tape. test c(MT, N) in forward direction, [TC].
936	**ETP N·**	0.45 + Ai ; 220	Erase Tape. erase one block length of (MT, N) in forward direction.
950	**JTG ·J A**	0.45 + Ai	Jump on Tape Good. if TC-ind. is off, #JA → c(LC), or if on, normal seq.
952	**JTE NJ A**	0.45 + Ai	Jump by Tape End. if No. N TE-ind. is on, #JA → c(LC), or if off, normal seq.

TABLE 1. (Continued)

Notes

The operation time is unchanged whether or not the modification of the address of an instruction takes place. The time for the modification is included in the operation time.
For the instructions which permit concurrent operations, the time needed for the central processing unit is written first, and the time needed for the output unit or the magnetic tape control unit second and so on.

Symbols and Abbreviations

Ai : instruction access time.
Aa : data access time.
→ : the replacement.
: the modification of the address of an instruction by the index registers, e.g. #IJA ≡ c(I)+c(J)+A (in modulo 10,000).
& : logical AND.
U : logical OR.
— : logical NOT.
rnd . round off.
Rem : remainder.
P-ind. : if the P-ind. is on, the address part of the next instruction in sequence is modified by the result in c(OR)ad of the former instruction, or if off, normal.
R-ind. : remainder indicator.
TC-ind. : magnetic tape check indicator.
[P], [R] or [TC] : the type of an instruction which affects P, R or TC-indicator.
c(X) : the contents of the register X, e.g. c(AC), c(#IJA), etc.
c(I), c(J) or c(H) : the contents of the index register IR I, J or H.
c(Y)ad : the contents of the address part of Y.
c(AC)pUA : the contents of the AC, on the UA part of which the operation is made.
c(UA)v : the content of the 12-th or the overflow digit of the UA.
c(E) & c(MD)odd : the extracted c(E) by the odd numbers in each digit of the MD.
c(CB) : the contents of the core buffer memory.
UAad : the address part of the UA.
UAm : the 11-th or the most significant digit of the UA.
(#IJn)i : the number of the i-th digit of #IJn.
loc($\pi/2$) : the location of a data word whose contents are $\pi/2$.
fx(x) : a fixed-point number x, e.g. fx(π).
fl(y) : a floating-point number y, e.g. fl(π).
Q#JA : the address in the quick access band No. Q which satisfies Q#JA = #JA (in modulo 50).
c(UA, i) : the contents of 11 bits specified by the weight i of the UA excluding the UA overflow digit.
(MT, N) : magnetic tape in No. N magnetic tape handler.
c(MT, N) : a block on (MT, N)
c(MT, N ; #JA) : a block whose block number is #JA on (MT, N).

3.3 Main Features of the Instructions

Features of the instructions can be summarized as follows:-

(1) The Clear Operation: The numerical code of any instruction has an even number except that of FMP/ (221) and FDC/ (223). By adding one to any even numbered code, it is possible to clear the contents of the AC just before the operation of the instruction of an even numbered code.

(2) Fixed-Point Operations: The double length fixed-point addition and subtraction operations.

(3) Floating-Point Operations : The double length floating-point addition and subtraction operations.

(4) Conversion Operations: A fixed-point number is converted into a floating-point number and vice versa.

(5) Sign Part Operations: The sign of the AC can be set plus or can be changed.

(6) Shifting Operations: Six various kinds of shifting; UA right shift, UA left shift, UA-LA right shift, UA-LA left shift, UA-LA left cyclic shift and head zero count and shift of UA-LA.

(7) Word Transfer Operations: The lower half of a double length mantissa in the AC can be stored as a floating point number with a compensating characteristic.

(8) Address Transfer Operations: Together with the address part addition and subtraction operations and index operations, the address calculation and counting are greatly facilitated by these operations. The c(IC) can be stored in the address part of a storage register.

(9) Block Transfer Operations: The instructions transfer a group of words which consists of fifty words or less between portions of the memory whose access time is different, i.e., between the normal and the quick access memory of the unit, between the drum memory and the core memory, or between the core memory and the magnetic tape memory.

(10) Control Operations: Fourteen kinds of control operation. Since the AC is a double length register, the state of not being zero should be expressed in two ways i.e. "the c(UA) are not zero" and "the c(AC) are not zero". Two control instructions which can test each of the above states are employed (JUN and JNZ). The instruction JSW can test the state of switches on the operator's console. The instruction JSX is particularly useful for the linkage of subroutine since the c(LC) is stored in one of the IRs and at the same time the control jumps to a specified location.

The instruction CMP is designed to perform the following operation;- the c(MD) are compared with the c(E) (E = the specified location). If the c(MD) is greater than the c(E), the control is in normal sequence, if equal, skips the next instruction, or if smaller, skips the next two instructions and proceeds from there. Furthermore this instruction can be used for normalized floating-point numbers as well as for fixed-point numbers, since the characteristic of a floating-point number is placed at the head of a word and a mantissa is placed next to it.

(11) Index Operations: The c(IR) can be replaced, increased, decreased or stored. Brief functions of the IRs are;-the direct indexing by the IRs, while the indirect indexing is performed by utilizing the instruction PSX. The modification of the address part of an instruction can be done twice by the c(IR) and/or c(LC). The instructions concerning the index operations can be also modified by the c(IR) or c(LC).

(12) Logical Operations: Since these operations have interesting properties, full details will be given later. Brief operations are: digitwise extract, bitwise extract, AND, OR and NOT operations.

(13) Special Operation: New instructions which are difficult to classify have been gathered, and since these instructions exhibit also interesting properties, details are fully described later.

(14) Input-Output Operations and Tape Control Codes: Selection of one or

several combinations of various I/O components. Alphanumerical and numerical I/O instructions and a floating-point form output instructions. A single read-in instruction RIN can read an instruction, a fixed-point or a floating-point number by using some special characters on paper tape, which are called Tape Control Codes.

(15) Magnetic Tape Operations: Briefly the instructions perform: writing, reading, testing, erasing, rewinding and examining the tape end and the parity error. By the operations, the magnetic tape is moved forward or backward, or is rewound, or is not moved. Any four-digit block number can be written. The block number can be used for the search and the confirmation of a block. Defective portions of the magnetic tape can be detected and skipped by suitable programming.

3.4 Details of Logical and Special Operations

Since the instructions concerning the logical and special operations have various interesting properties, the full details of these instructions are described. The same symbols and abbreviations used in the previous sections will be used.

Logical Operations

The following instructions are described; ERE, AND, IOR, NOT, WAN, WOR.

The instruction EAD is explained in the Fixed Point Arithmetic Operations. (in Appendix B)

The logical operation is made by regarding a zero or one of a binary number, four of which can constitute a binary coded decimal number (BCD code), as a false or true value of a logical variable respectively. The truth tables for the logical AND, OR and NOT (whose symbols are &, U, $\bar{}$) are as follows;

x	y	x & y	x U y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

x	$\bar{x}$
0	1
1	0

Each bit has a weight in BCD code, i.e. 1, 2, 4 or 8 and a bit is called a bit of the weight 1, 2, 4 or 8 respectively. A particular bit in a register is specified by a number of the digit position and a number of the weight.

The symbol c(UA,i) is used to show the contents of 11 bits specified by the weight i of the UA excluding the AC overflow digit. The c(MD) can be used as a digit-by-digit extractor for ERE (and EAD), or as a bit-by-bit extractor for AND, IOR and NOT. Numbers in a particular weight can be extracted by WAN or WOR.

502 ERE IJ A "Extract and Replace" (0.35 + Ai + Aa)

c(UA) & c(MD)even U c(E) & c(MD)odd → c(UA).

The least significant digit of the UA as well as of the register of loc. E (=#IJA) corresponds to that of the MD, the second digit to the second, and so on.

A decimal number in each digit of the UA whose corresponding digit of the MD contains an odd number is replaced by a decimal number of the corresponding digit of the register of loc. E, and remains as it is if the corresponding digit of the MD contains an even number. The c(MD) are in this

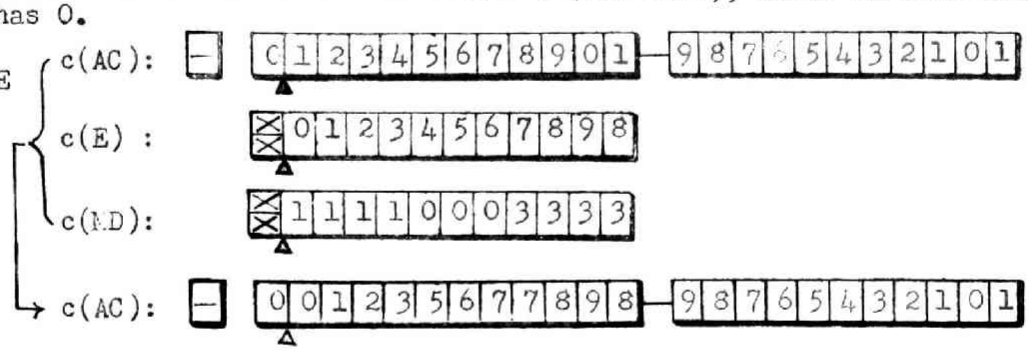
sense a digit-by-digit extractor.

N.B.1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA), c(MD) and c(E) are unchanged.

2. Both a sign and a number in the overflow bit of the MD are neglected.

3. An odd number has 1 in the weight 1 (BCD code), while an even number has 0.

e.g. ERE



520 AND IJ A "AND" (0.35 + Ai + Aa)

$\{c(UA)\ \&\ c(E)\}\ \&\ c(MD)\ U\ c(UA)\ \&\ \overline{c(MD)} \rightarrow c(UA);\ 0 \rightarrow c(UA, 8).$

The AND operation is made only between a binary number in each of the particular bits of the UA whose corresponding bits of the MD contain 1's and a binary number in each corresponding bit of the register of loc. E (=#IJA). The result is placed in each of the corresponding bits of the UA. The contents of bits of the UA other than the above remain unchanged, except that numbers in the weight 8 of the UA are made zeros. The c(MD) are in this sense a bit-by-bit extractor.

N.B.1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA), c(MD) and c(E) are unchanged.

522 IOR IJ A "Inclusive OR" (0.35 + Ai + Aa)

$\{c(UA)\ U\ c(E)\}\ \&\ c(MD)\ U\ c(UA)\ \&\ \overline{c(MD)} \rightarrow c(UA);\ 0 \rightarrow c(UA, 8).$

The OR operation is made only between a binary number in each of the particular bits of the UA whose corresponding bits of the MD contain 1s and a binary number in each of the corresponding bits of the register of loc. E (=#IJA). The result is placed in each of the corresponding bits of the UA. The contents of bits of the UA other than the above remain unchanged, except that numbers in the weight 8 of the UA are made zeros. The c(MD) are in

this sense a bit-by-bit extractor.

N.B.1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA), c(MD) and c(E) are unchanged.

2. The following formulae are equivalent to the above; c(UA) U c(E) & c(MD) → c(UA); 0 → c(UA, 8).

516 NOT .. - "NOT" (0.35 + Ai)

$\overline{c(UA)}$ & c(MD) U c(UA) & $\overline{c(MD)}$ → c(UA); 0 → c(UA, 8).

The NOT operation is made only on a binary number in each of the particular bits of the UA whose corresponding bits of the MD contain 1's, and the result is placed in the same bits of the UA. The contents of the bits of the UA other than the above remain unchanged, except that numbers in the weight 8 of the UA are made zeros. The c(MD) are in this sense a bit-by-bit extractor.

N.B.1. The AC sign and a number in the AC overflow digit are unchanged. The c(LA) and the c(MD) are unchanged.

2. Numbers in the index and the address part of this instruction are neglected in the operation.

3. In other words, the above operation is the "Exclusive OR" between the c(UA) and the c(MD).

550 WAN IJ n "Weighted AND" (0.35 + Ai)

c(UA, 1) & c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).

The AND operation is made between a binary number in each of the bits of the weight 1 and a binary number in each of the corresponding bits of the weight specified by #IJn of the UA, and the result is placed in the same bits of the weight 1 of the UA. Numbers in the weight 2, 4 and 8 of the UA are reset to zeros at the end of the operation.

N.B.1. The AC sign and a number in the AC overflow digit are unchanged.

2. The #IJn should be ordinarily 2, 4 or 8 which corresponds to the weight 2, 4 or 8.

3. If #IJn = 0 or 1, the c(UA)m ... 1 are reset to zeros.
If #IJn = 3, 5 or 9, equivalent to #IJn = 2, 4 or 8 respectively.
If #IJn = 6 or 7, the result is equivalent to the OR between the results of WAN 00 2 and WAN 00 4.

552 WOR IJ n "WeightedOR" (0.35 + Ai)

c(UA, 1) U c(UA, #IJn) → c(UA, 1); 0 → c(UA, 2, 4, 8).

The OR operation is made between a binary number in each of the bits of the weight 1 and a binary number in each of the corresponding bits of the weight specified by #IJn of the UA and the result is placed in the same bits of the weight 1 of the UA. Numbers in the weight 2, 4 and 8 of the UA are reset to zeros at the end of the operation.

N.B.1. The AC sign and a number in the AC overflow digit are unchanged.

2. The #IJn should be ordinarily 2, 4 or 8 which corresponds to the weight 2, 4 or 8.

3. If #IJn = 0 or 1, the c(UA)m...1 are reset to zeros.
If #IJn = 3, 5 or 9, equivalent to #IJn = 2, 4 or 8 respectively.
If #IJn = 6 or 7, the result is equivalent to the OR between the result of WOR 00 2 and WOR 00 4.

Special Operations

The following instructions are described; PSX, SCT, TLU. They perform rather complex operations and moreover as a result of the operations they modify the address part of the next instruction in sequence when the next instruction is read. To distinguish these situations, the P-indicator (also P-indicator lamp on the console) is set on. The P-indicator is set off after the modification of the address part of an instruction and at the beginning of the execution of all kinds of instructions.

860 PSX IJ A "Set Pseudo Index" (0.35 + Ai + Aa)

Numbers in the address part of the register of loc. #IJA are read and reserved in the address part of the OR for the modification of the address part of the next instruction, and for this purpose the P-indicator is set on.

N.B.1. Numbers which modify the next instruction are not #IJA, but the address part of its contents. This is just the same situation as where the number of an index register is referred to, but its contents are used for the modification.

e.g. 1. PSX IJ A
ADD I'J' A'

The effective address E for ADD; E ≡ c(#IJA)ad + #I'J'A' (in modulo 10,000).

e.g. 2 PSX IJ A
JXR HJ' A'

If c(H) ≠ 0, the control jumps to E ≡ c(#IJA)ad + #J'A' (in modulo 10,000).

518 SCT .. - "Shift and Count" (1.4av + Ai)

The c(AC) are shifted to the left until a non-zero number appears in the m position of the UA. The number of places shifted is counted and reserved in the address part of the OR for the modification of the address part of the next instruction, and for this purpose the P-indicator is set on.

N.B.1. The c(UA)v is reset to zero at the beginning of the operation.

2. If c(UA)m $\neq$ 0, it should be regarded as a zero-place shift.

3. If the c(AC) except the c(UA)v are all zeros, the 22-place shift takes place, so 22 is used for the modification.

4. The AC sign is unchanged.

5. Numbers in the index and the address part of the instruction are neglected in the operation.

6. This instruction is especially useful for the heading-zero suppression of the output routine, or for the search of positions of a certain pattern.

360 TLU MJ A "Table Look Up"

The $|c(MD)|$ (the absolute value) are compared with the $|c(E)|$, ..., $|c(E + i$ ..., $|c(BE)|$ successively ($E = \#JA$, $BE \equiv 199$ in modulo 200, $0 \leq BE - E < 200$, $i = 0, 1, \ldots, BE - E$). If the condition, $|c(E + i)| \geq |c(MD)|$, is detected for the first time, this $E + i$ (not the $c(E + i)$) is reserved in the address part of the OR for the modification of the address part of the next instruction, and for this purpose the P-indicator is set on. The loc. E must be in the normal access memory of the drum i.e. $0 \leq E \leq 3999$.

If the above mentioned condition is not detected, the control jumps to the c(E). At this time the P-indicator is not on.

N.B.1. This instruction can be used for both fixed-point and normalized floating-point numbers.

2. The contents of any register , including the c(AC), c(MD), c(E), ..., c(BE) are unchanged.

3. For comparison, the fixed point subtraction $|c(E + i)| - |c(MD)|$ is made (at the MQ). The sign of the difference decides the above condition.

4. Special care should be taken for the inclusion of a number in the overflow bit of the MD as well as of the register of loc. E in the subtraction. Thus $|c(MD)|$ or $|c(E + i)| < 2$. This makes possible the comparison of two normalized floating-point numbers by the fixed-point subtraction.

5. A table should consist of 200 words or less, and should be stored in a band of the normal access memory of the drum (whose locations are; 0 - 199, 200 - 399, ..., or 3800 - 3999; 20 bands).

6. In a table, words should be ordinarily arranged in ascending

order of values.

7. The M should be 1, 2 or 3 which corresponds to the IR1, 2 or 3. If M = 4, ..., 9, 0 and moreover it is not the case the P-indicator is on, then the operation is equal to NOP.

e.g.1. A word in a table contains an argument x in its upper half, and f(x) in its lower half.* (An argument x is calculated and is placed in the MD for comparison); * The location of the table begins from #JA.

```
TLU MJ A      loc (x) → c(OR)ad
ADD/00 0      x, f(x) → c(UA)
```

Both x and f(x) are read and placed in the UA.

2. A table of arguments x and a table of functions f(x) are made separately. The location begins from #JA for the arguments, and from #JA + B for the functions;

```
TLU MJ A      loc (X) → c(OR)ad
ADD/00 B      c(loc (x) + B) = f(x) → c(UA).
```

3. A word in a table contains an argument x at its upper half, and the location of f(x) in its address part.

```
TLU MJ A      loc (x) → c(OR)ad
PSX 00 0      loc (f(x)) → c(OR)ad
ADD/00 0      f(x) → c(UA)
```

The f(x) is read and placed in the UA.

4.

```
TLU MJ A
RAA/00 0      loc (x) → c(UA)ad
```

The loc (x) is read and replaces the c(UA)ad.

5.

```
TLU MJ A
SEX H0 0      loc (x) → c(H)
```

The loc (x) is read and replaces the c(H).

Chapter 4

MAGNETIC TAPE SYSTEM OF THE COMPUTER

4.1 Introduction

In this chapter, the details of the magnetic tape system which is connected to the main part of the Computer are presented with emphasis on design features and problems (Ya 3). A brief description of the subject, however, was given in a previous chapter in connection with the description of the computer (Ya 2,3).

The magnetic tape system as well as the central processing unit was developed and built in collaboration with Hitachi, Ltd. The author contributed mainly to the system design and the logical design of the magnetic tape system, while the constructin of the system was made by the manufacturers. The author also co-operated in the adjustment and testing of the system, and is now helping in the maintenance of the system.

The magnetic tape of the KDC-I is used for the storage of intermediate results and is also used for permanent storage of progrmas and data. At present the magnetic tape cannot be used as an input/output medium. The system includes magnetic tape hadlers and a magnetic tape control unit which contains a high speed magnetic core memory. Up to four magnetic tape handlers can be connected to the magnetic tape control unit; this unit is connected to the main part of the computer. At present two magnetic tape handlers are connected and utilized (Ki 2).

The magnetic tape system has been attached to the main part of the computer since October 1960. The adjustment of the tape system consumed almost three months owing to lack of experience until various magnetic tape test programs produced by the Kyoto University Programming Group (Ku 4) were operated correctly. Thus the adjustment ended in December 1960. However, some faults were found afterwards and were corrected in May 1961.

Since then the magnetic tape system has been in constant use. The high speed magnetic core memory in the tape control unit is fully utilized since the memory of the main part of the computer is a medium speed magnetic drum memory.

The system design of the tape system was carried out as a part of the design of the whole system. The logical design of the tape system was completed by October 1959. Owing to lack of experience and because the manufacture of the system should have been officially completed by March 1960, the functions of the system were not made too complex. Nevertheless various trials were made. The main features of the tape system can be said to be the concurrent operation of the tape and of the main part of the computer, a new block number system which was named a variable block number system (Ya 4) and an instruction system which enables a simple treatment of the drop-out problem of magnetic tape.

The magnetic tape system of the KDC-I is believed, so far as the author is aware, to be the first successfully operating tape system in a domestic computer. In Fig. 4.1 a view of the magnetic tape system of the KDC-I is shown.

The magnetic tape system of the KDC-I is at present a secondary memory. Since the output capability of the computer is weak, it is strongly hoped that the magnetic tape can serve as an input/output medium. However, this plan has not been realized mainly for economic reasons.

The following abbreviations will often be used in this chapter:

CPU : Central Processing Unit or Main Part of the Computer.
TCU : Magnetic Tape Control Unit.
MTH : Magnetic Tape Handler.
MT : Magnetic Tape.
CB : Core Buffer or Magnetic Core Matrix Memory.
LP : Loadpoint of the Magnetic Tape.
TE : Tape End.
TC : Tape (Parity) Check.

Fig.4.1-View of the magnetic tape system of the KDC-I.
From left to right; the central processing unit and the operator's console, the tape control unit, and the two magnetic tape handlers.

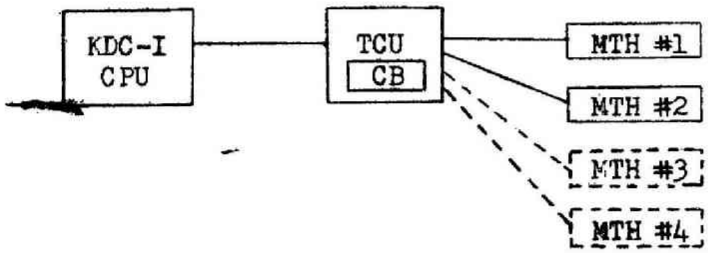


Fig.4.2-Block diagram of the magnetic tape system.
Dotted lines show the units scheduled for installation.
CPU: Central Processing Unit, TCU: Tape Control Unit,
CB: Core Buffer, MTH: Magnetic Tape Handler.

4.2 Organization of the Magnetic Tape System

Various tape systems were suggested. However, the following system was proposed and constructed. The main points will be described in turn.

(1) The use of the magnetic tape (MT) whose width is ca. 12.7 mm (1/2 inches), with length approx. 1,100 m (3,600 feet), and which is capable of storing as many as 7,000 50-word blocks.

(2) The use of the magnetic tape handlers (MTH) whose magnetic tape operation speed is ca. 150 cm/sec (60 inches/sec) in both forward and backward directions, and whose magnetic tape starting and stopping time is approx. 7 ms. The MTH has an 8-channel read/write head and an erase head which is situated approx. 6 mm apart from the read/write head. The main controlling mechanism is composed of an electro-mechanical control mechanism and a pneumatic one. Up to four MTHs may be connected to the computer; two of them were actually made.

(3) The biggest problem with regard to the design was the use of MT as an input/output medium. This plan has not been realized mainly for economic reasons.

(4) The next important problem was the selectin of the type of a buffer storage. Two plans were suggested. One was the use of a portion of the drum memory for buffer storage. The other was the use of the magnetic core matrix memory. The former plan was apparently more economical than the latter. Nevertheless the latter plan has an important merit. Since the instruction system of the computer is one-and-a-half address system and the memory of the main part of the computer is a medium speed magnetic drum, the problem of the access time of the memory was very serious, even though a delay-line type quick access memory was made. Thus by installing a high speed core matrix memory, even if the capacity were small, the processing speed of the computer could be expected to become much faster than in the case of the former plan. The latter plan was finally accepted, and a 50-word core matrix was decided

upon for construction. The capacity of 50 words is almost a minimum feasible limit for this type of computer and was chosen mainly for economic reasons.

(5) The block diagram of the system was thus decided upon to be the one shown in Fig. 4.2 Up to four MTHs can be connected to the TCU, which is then connected to the main part of the computer. At present only two MTHs are connected. The TCU contains a 50-word magnetic core matrix buffer storage or core buffer (CB).

(6) The CB is used not only as a buffer storage, but also as high speed storage registers, whose addresses are from 4,200 to 4,249, and whose access time is 50 μs.

(7) The concurrent operation of the magnetic tape and of the main part of the computer should be possible. Thus the processing speed is expected to increase.

(8) The rewind operation should be done concurrently with other tape operations.

(9) The magnetic tape code should be the same as the paper tape code, though for the time being only numerical characters are used. Thus the design of various off-line units is expected to become simpler.

(10) Information is written on MT as a 50-word block mainly bacause the capacity of the core buffer is 50 words.

(11) To ensure the reliability of the magnetic tape system, an odd parity bit is added to each character code (in the case of the magnetic tape system the sprocket is counted as a bit while in case of paper tape it is not counted) and an even parity bit is added to the end of each channel of a block. The former will be called a character parity, and the latter a channel parity.

In case of reading, both parity bits are examined. If a parity error is detected, a Tape Check indicator (TC-indicator) is turned on, but the computation does not stop. The state of the TC-indicator can be examined by an instruction. Thus by programming various treatments for this situation

become possible, e.g., try to read once again, or erase that portion of MT and so on.

In the TCU, the parity and validity checks are made. In the core buffer, every digit has an even parity bit. They are examined when reading and writing.

(12) A new block number system which is named a variable block number system was adopted. Any four-digit block number can be written at the head of a 50-word block during the writing operation on the magnetic tape. Thus the block number is not fixed on the MT beforehand like other systems. Flexibility concerning the use of the block number is expected to increase.

(13) Nine computer instructions are added for the operation of magnetic tapes. Two more instructions are also made for the block transfer of information between the core memory and the drum memory. Briefly the operations of these tape instructions are: Writing, reading, testing, erasing, rewinding, and detecting the tape end and parity error. Under this instruction system, it is possible to avoid defective portions of the magnetic tape, to detect the file end by using the block number, and so on, by suitable programming.

(14) The control panel is attached to the TCU, which is used for the local supervision of the operations of the TCU and of the MTHs, the TCU check and local adjustment, though most of the magnetic tape operations can be supervised at the operator's console.

4.3 Magnetic Tape (MT)

Various length of MT whose width is 12.7mm(1/2") can be used. Both ends of the MT are perforated for the purpose of sensing the load point and the tape end. The speed of the MT is 150cm/sec in case of reading and writing. On the MT, a 50-word block is written as a unit. The block length on the MT is ca. 11cm. The inter-block gap is ca. 4cm. The read/write head is expected to be situated in the middle of the inter-block gap when the MT is stopped (2 ± 0.2cm from the end of a block or from the beginning of a block).

A standard reel for the MTH contains 1,100m(3,600 ft.)of MT, which can store as many as 7,000 blocks or 350,000 words.

1. Magnetic Tape Character Coding

The same codes as for paper tape are used; however, the arrangement and the variety of codes used are different. The position of the character parity is at the edge of the MT. All these codes are written magnetically by the instruction BTP. Since the magnetic tape is used only as an auxiliary memory, not as an input/output at present, 16 varieties of codes are used.

4.3 In Fig. 4.3 the magnetic tape character coding is shown.

2. Layout of a Block on the Magnetic Tape

A 50-word block is written on the MT by the operation of the instruction

4.4 BTP. A brief drawing of the layout of a block on the MT is shown in Fig.4.4 The information is written in the following way:

(1) four block beginning codes.

(2) one no-effect code.

(3) four numerical codes as a 4-digit block number.

(4) one no-effect code.

4.5 (5) 50 words of information, the arrangement of which is shown in Fig. 4.5.

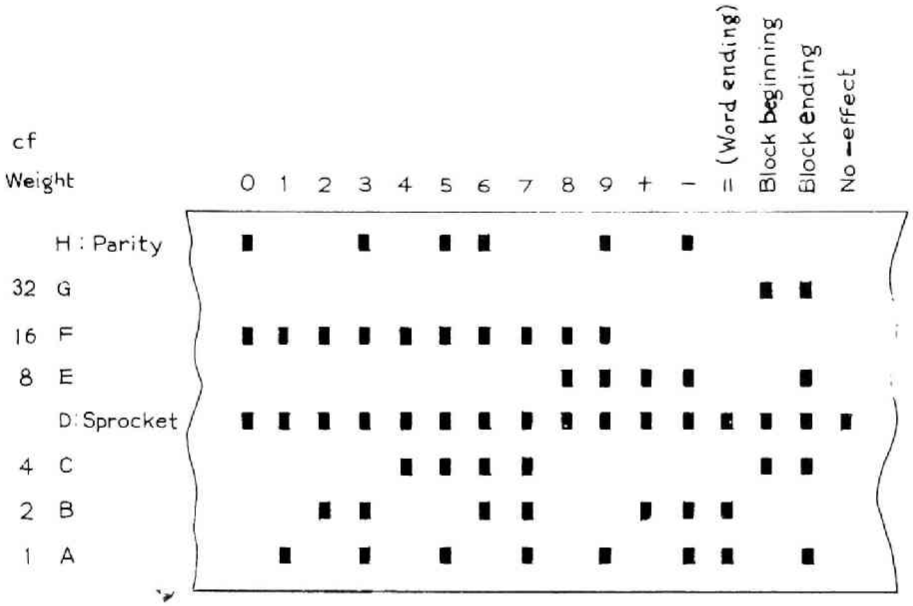


Fig.4.3-Magnetic tape character coding.

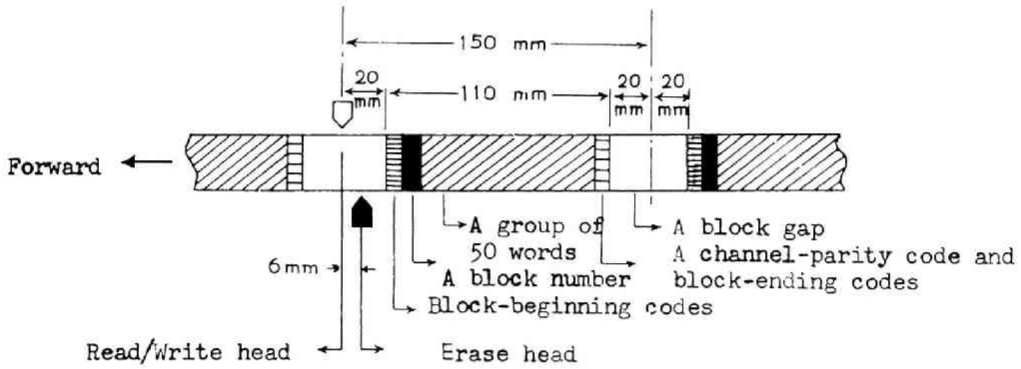


Fig.4.4-Layout of a block on the magnetic tape.

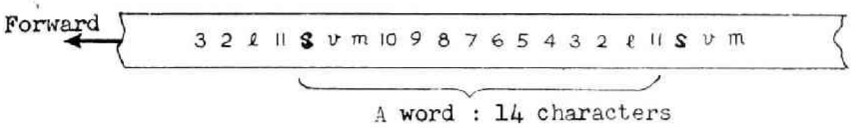


Fig.4.5-Layout of a word in the block.
s: Sign bit, v: Overflow bit,
m: 11th or most significant digit,
1: 1st or least significant digit,
": Word-ending mark.

(each word is terminated by the word ending mark ").

(6) one no-effect code.

(7) a channel parity code; in each channel, the number of 1's is made even by this code.

(8) after these seven kinds of codes, four block ending codes are written at the end of a block.

The block beginning codes and the block ending codes are used for controlling the movement of the MT.

The block number and the sprocket bits are not written beforehand, but are written at the time of the operation of the instruction BTP. Any 4-digit block number can be written. The block number is used for the search and the confirmation of a block in programming.

3. Read and Write Operations

An 8-bit code of each character is written simultaneously through an 8-channel read/write head of an MTH with a density of 64 characters per cm by the biased NRZ (Non-Return-to-Zero) method. The principle of this method
-6 is illustrated in Fig. 4.6. The erase head erases the MT and sets its magnetic flux - Ø; the current in the read/write head flows as shown in Fig 4.6 (1). Then the magnetic flux of the MT becomes as shown in Fig. 4.6 (2); the flux is changed only when "1" is written, and is unchanged when"0" is written.

When reading information from the MT, the derivative of the flux is detected at the read/write head, the polarity is rearranged, and the wave-
-7 form is reshaped, as shown in Fig. 4.7.

The information density per channel is thus 6.4 bits / mm, or 3.2 pulses/mm, while the speed of a read/write operation is 9,600 characters/sec. Since these operations are performed through one read/write head and one erase head, simultaneous read and write operations are not possible.

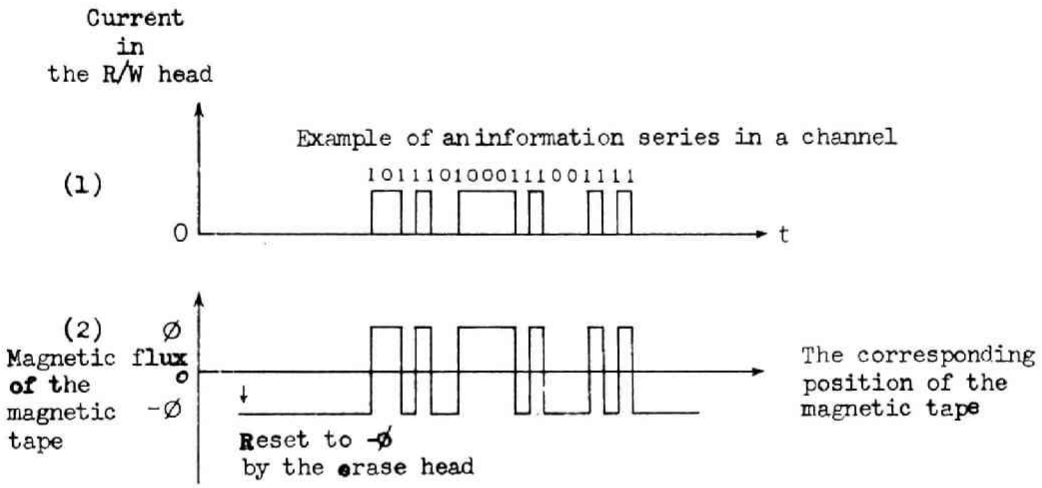


Fig.4.6-Principle of the write operation.

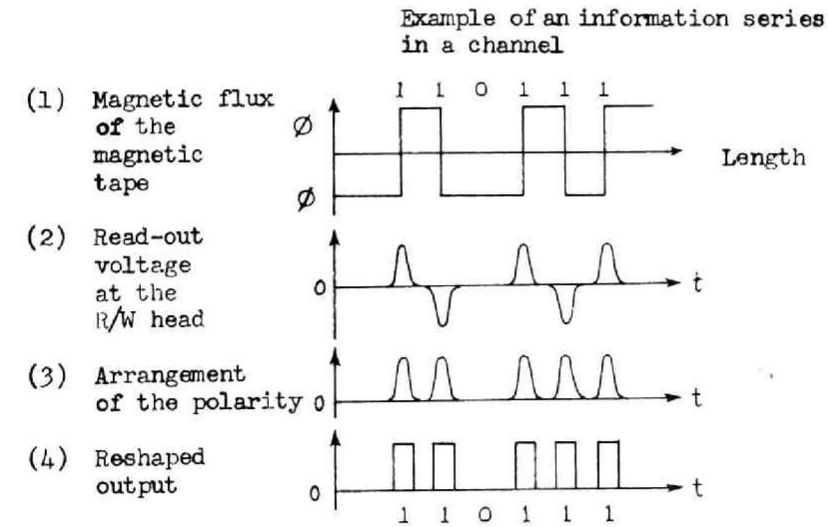


Fig.4.7-Principle of the read operation.

4.4 Magnetic Tape Handlers (MTHs)

An MTH is a mechanical device in which a reel of MT is placed and used. The main parts of the MTH are a controlling mechanism, a read/write head, an erase head, a tape-end (TE) detector, a load-point (LP) detector, the file protector, and the ready indicator. The controlling mechanism is composed of electro-mechanical and pneumatic control mechanisms.

The position of the MT is shown schematically in Fig. 4.8.

The movement of the MT is controlled automatically by the computer, but if necessary it can be controlled manually at the MTH. Both ends of the MT are perforated for the purpose of sensing the tape-end and the load-point of the MT, and are detected by the TE or LP detector, and if detected, the movement of the MT is automatically stopped in any case.

In either forward or backward motion of the MT, the MT is driven at the constant speed of 150 cm/sec except for a few milliseconds while starting and stopping. In the case of the rewind motion of the MT, the MT is rewound to its load point, and the motion cannot be interrupted if once initiated. The time needed for the rewind depends upon the length of the MT to be rewound, but roughly, the average speed of the rewind is roughly assumed to be three times faster than the ordinary 150cm/sec.

4.5 Magnetic Tape Control Unit (TCU)

All magnetic tape operations are controlled mainly by the TCU. The magnetic core memory in the TCU is used as a high speed memory whose location are from 4,200 to 4,249, and also as a buffer storage for reading and writing operations of the magnetic tape.

If an instruction concerning the magnetic tape operation is decoded by the main part of the computer, it will be transmitted to the TCU, and the execution begins under the control of the TCU; the computer takes the next instruction in sequence and proceeds from there concurrently if the

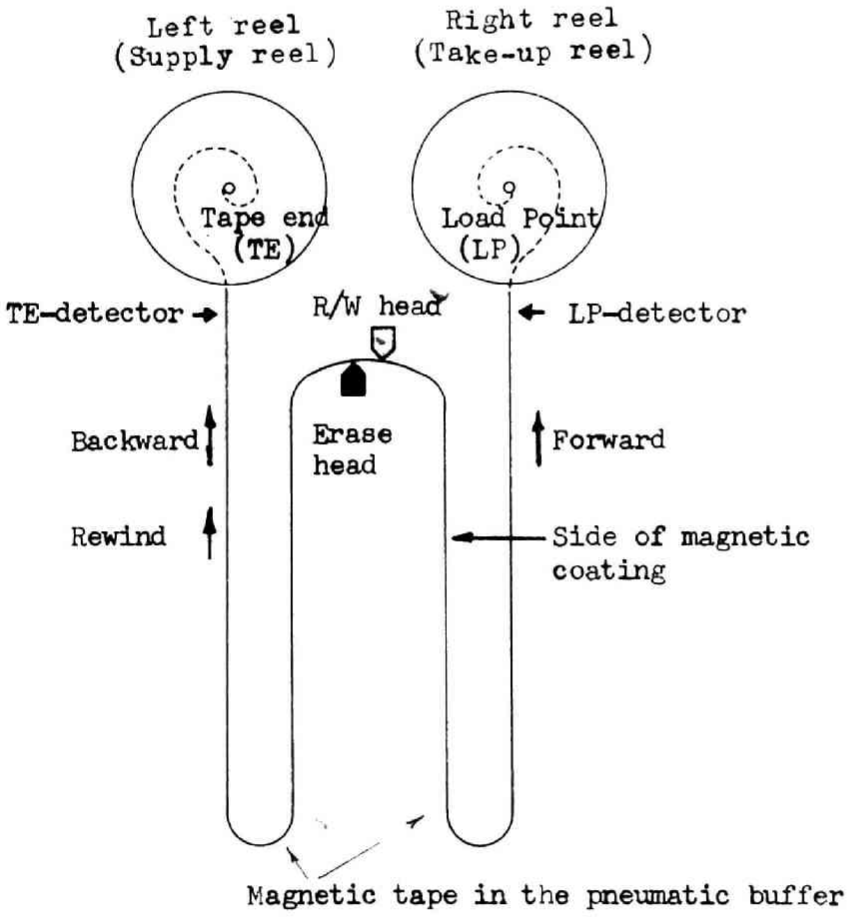


Fig.4.8-Position of the magnetic tape in the MTH mechanism (A front view).

instruction does not refer to the TCU or the MTH under operation. If this is not the case, the execution of the next instruction is delayed until the execution of the former instruction finishes. Therefore by suitable programming it is possible to increase the processing speed. In programming, however, instructions should be regarded as executed one after another in sequence.

In Fig. 4.9 a block diagram of the TCU is shown. Each small block in the diagram is classified according to the function it performs. Thus the name which is given to each block represents one of the following: the name of a function unit, a register or an indicator, or the function of a block.

The details of the TCU are now explained; most of the functions of the blocks are also described.

(1) The same type of transistorized dynamic flip-flop circuit used for the CPU is also used for most of the circuits in the TCU, because it is simple to connect the TCU to the CPU, the logical design can be done in the same way, and moreover there was little time available for developing a suitable static circuit. However it was foreseen that the use of static circuits for the whole TCU would have simplified the logic circuits.

(2) The CB has the capacity of 50 words with 50μs (or one-word time) access time. One word (or 12 decimal digits) has 60 bits since each digit has a parity bit. The actual CB is of a current coincidence type and consists of 100 30-parallel-bit words with 17μs access time, and for 60 bits of information two cycles are used. The structure of the core matrix is of 4 × 25 (address) × 30 (parallel bits), with five 4 × 25 matrices constructed in each plane; thus six planes are used in all. The S-1 type ferrite cores of the General Ceramic Co. were used since domestic cores did not fully satisfy requirements at the time of the design.

(3) The selection of a particular MTH out of the four MTHs is made by using reed relays whose operation time is of the order of a few milliseconds.

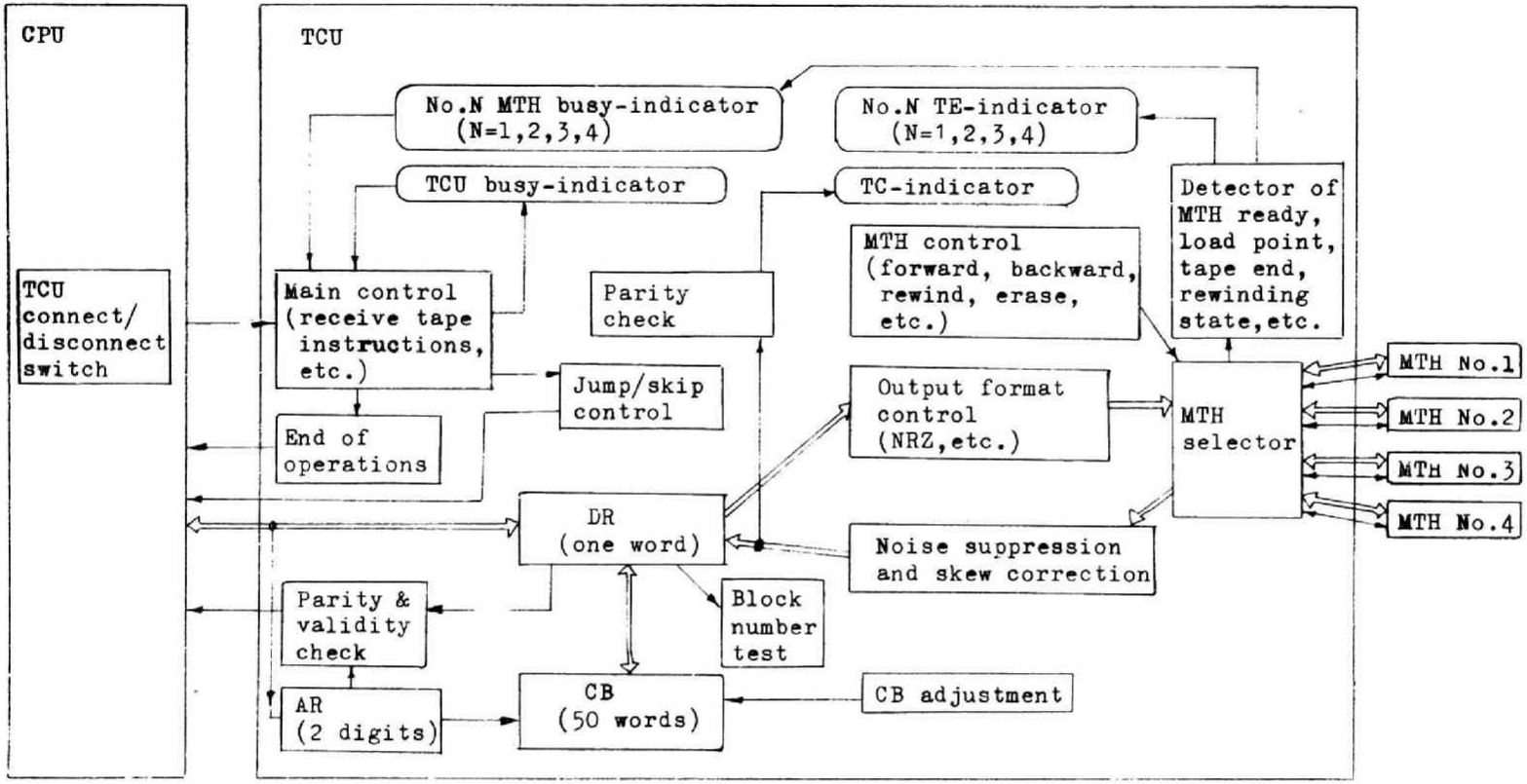


Fig.4.9-Block diagram of the TCU of the KDC-I.
CPU: Central Processing Unit, TCU: Magnetic Tape Control Unit, MTH: Magnetic Tape Handler, CB: Core Buffer, DR: Distributor Register, AR: Core Address Register, TE: Tape End, NRZ: Non-Return-to-Zero form of output signal.
⇔: Information, ↔ : Control signals.

At present there is no problem concerning the noise and life of the relays. However, the use of an electronic switch would be better from the standpoint of the above mentioned factors though the use of mechanical relays for this circuit are more economical even at present.

(4) There are two important registers in the TCU, i.e.,a Distributor Register (DR) and a Core Address Resister (AR). All information read from or written to the magnetic core memory or the magnetic tape memory is stored temporarily in the DR. The DR has 12 digits, the capacity of a word. Moreover each digit has a parity bit. The main function of the DR are the series-parallel conversion and the buffering action between the tape and the CB. The even parity and the validity of the code are examined at the DR, and since all information flows in the DR, error checking for the operation of the CB is made by this checking circuit.

The AR is used to specify the location of the core memory. Since it is used only for the 50-word core memory, the AR has a 2-digit capacity, and the two least significant digits of the core location are stored in the AR. The validity checking is also made here.

(5) The output format is controlled in the TCU. The NRZ (Non-Return -to-Zero) signal is generated by flip-flops in the TCU, transmitted to the MTH, amplified and reshaped in the MTH, and written on the MT.

The even channel parity code at the end of a block is made by resetting the final state of the flip-flops since the flip-flops are modulo 2 counters. The odd character parity on the even channel parity is realized by arranging the number of characters in a block. All these check codes are examined in the case of reading. Characters are written at the rate of 9,600 characters/sec., that is, in every 104 μs (in every two-word time) a character is written.

4.10 (6) In Fig. 4.10 the input circuit of a channel in the TCU is shown. There are eight such input circuits in all. The read-out signal from the read/

write head is amplified and reshaped in the MTH, and then transmitted to the TCU by a coaxial cable (RG 58 A/U and less than 10 meters long). After passing the relay MTH selector circuit, the signal is applied to the basic logic circuit of the computer. Then the duration of the signal or pulse is measured by the counter or the logic circuits SAi and SBi. If the pulse count is greater than 4 or the duration of the pulse at the output of the amplifier exceeds approx. 17.4μs , the pulse is recognized as a signal and temporarily stored in IRi, otherwise it will be recognized as a noise. The circuit is a sort of a digital integration circuit used for the suppression of noise. Since there would be a time displacement error among the timing of signals in each channel, and skewness of the MT would probably cause a great deal of error, the output signal from the sprocket channel which is in the center of the MT is used as a reference.

Since the signal comes approximately once in every 104μs , and the signal at the IBi must be read into the DR in synchronization with the timing of the DR, the signal temporarily stored in the IBi should be tranferned to another temporary storage TRi which functions as a synchronization buffer. The transfer timing BTR is given 50μs after the sprocket signal is detected at the output of the noise suppression circuit. Thus the permissible time displacement error is 40μs with reference to the sprocket signal. Moreover the permissible limit concerning the instantaneous compression of the timing of the signal is approx. 60μs . Any degree of expansion of the timing is permissible so far as the time desplacement error is within this range. The signal in the TRi is transferred into the DR with the necessary timing, say, Tx as is shown in the figure.

(7) Parity checking is made on all information which is utilized. For example in the case of the instruction BLS or Block Search, the character parity of all block numbers tested is examined in addition to the parity check of the object block. If parity error is detected in case of reading, the TC-indica-

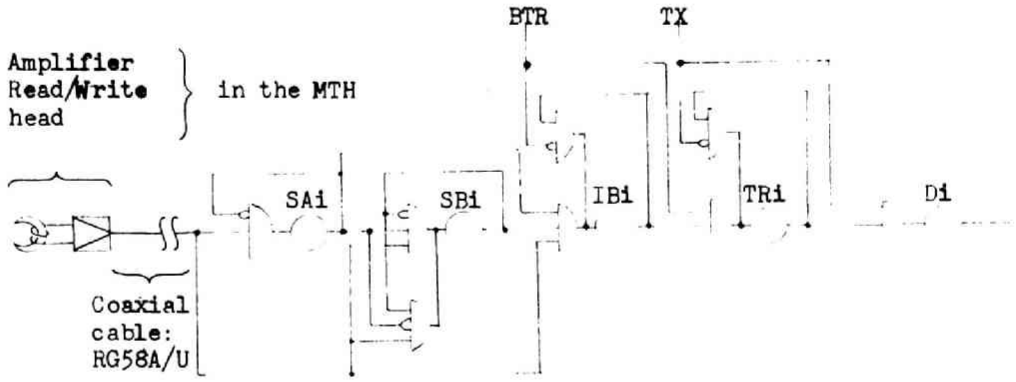


Fig. 4.10-Input circuit of a channel in the TCU.

tor is turned on, but the computation does not stop.

If parity or validity error is detected at the DR at the time of reading information from the MT, the TC-indicator is also turned on. Except in the above case, the error will cause the computation to stop.

The channel parity is examined by the same flip-flops which are used for making the NRZ output signal, i.e.,counting the input signal in modulo 2 fashion.

(8) The concurrent operation of the CPU and the TCU is performed by introducing two kinds of busy-indicators, i.e.,a TCU busy-indicator and No.N MTH busy-indicators (N=1,2,3,4). By most of the tape operations,both TCU and MTH busy-indicators are turned on. But in the case of rewinding the MT, only a particular MTH busy-indicator is turned on.

(9) Since the TCU has rather an independent control function, it was easy to add an adjustment circuit. In the main control circuit of the TCU there are several flip-flops which receive the tape instructions from the CPU. It was made possible to trigger these flip-flops manually at the control panel. Thus in case of local adjustment, most of the magnetic tape instructions can be executed by depressing the corresponding buttons on the panel.

The CB adjustment circuit was also attached to the TCU. The function of the circuit is to reset the contents of all core memory, read all contents repeatedly, write all zero, all ones, or else two of the very unfavorable patterns into the core memory (when reading, the S/N ratio is believed to become low).

Thus most of the adjustment of the CB and the tape operations can be done at the TCU control panel. This function of the TCU has actually been utilized since the TCU was constructed and it had to be adjusted without the CPU after the CPU was installed at the University.

4.6 Magnetic Tape Operations

The reliability of the MT was till an unknown factor at the time of the design. However it could easily be forseen that there would be some defective spots on the MT and that the number of such spots might increase during use.

Thus after writing, the information on the MT should be tested (parity checking), and, if error is detected, that portion should be erased.

Nine instructions are added to the computer for the operation of the MT. Two more instructions are also made for the block transfer of information between the core memory and the drum memory.

Block number

A new block number system was adopted in which any 4-digit block number can be written at the time of writing information on the MT. Since block number is not fixed on a particular portion of the MT and can be changed if necessary, this block number system was called a variable block number system.

This system can be applied in the following way:-

(1) Sequential numbering is possible.

(2) Index number of data or a number converted from the original index number is used as a block number. For example the storage of subroutines can be made in this way.

(3) The use of the MT as an addressable memory can be performed by writing suitable block numbers on the MT beforehand. The sequence of the numbers can be selected arbitrarily.

(4) The use of a certain number as a special control code is possible.

(5) A quick access memory can be realized by storing the same information with the same block number on several portions of the MT.

(6) If a portion of MT becomes defective, it is possible to rewrite the same information on another good portion of the MT without changing the block number.

Details of the operations of the magnetic tape instructions

The definition of the operations of the magnetic tape instructions is now given. The numerical code, the symbolic code, the title of an instruction, and the operation time in ms are written first. The instruction system of the KDC-I is one-and-a-half address, consisting of a 3-digit function part and a two-digit index part (which is used not only for the specification of an index register, but also used for the specification of the number of the MTH), and a 1-digit breakpoint part, and a 4-digit address part.

The operation time is unchanged whether or not the modification of the address of an instruction takes place. The time for the modification is included in the operation time. For the instructions which permit concurrent operation, the time needed for the CPU is written first, the time needed for the TCU second, and the time needed for the MTH third. Ai stands for the instruction access time, and Aa the data access time. The symbol # is used to show the modification of an address. Other symbols and abbreviations used are commonly accepted one, however, full details can be found in the previous chapter.

Then the definitions follow, and some notes are added if necessary.

910 BTP NJ A "Buffer to Tape" (0.45 + Ai; 220ms)

The erasure of the MT of the MTH specified by N begins, and the MT is started forward. A block number #JA and 50 words in the core memory are written as a block on the successively erased portion of the MT.

Next the MT is stopped and the erasure is also stopped.

N.B.1. The execution of the instruction is delayed until the TCU and the No.N MTH busy-indicators in the "on" state by previous instructions are restored to the "off" state.

2. After 0.45 ms from the beginning of the execution, the next instruction in sequence becomes operative concurrently. During the operation, the TCU and the No.N MTH busy-indicators are on, i.e., for 220ms.

3. If the No.N TE-indicator has been on, the operation is equivalent to NOP. If the No.N TE-indicator is set on during the write operation, the MT stops only after finishing the operation.

4. If the MT starts from its load point, the writing of information begins after the MT leaves the load point.

936 ETP N. - "Erase Tape" (0.45 + Ai; 220ms)

This instruction erases the MT for the length of a block and an inter-block gap.

N.B.1. The same operation as that of BTP is done except that this instruction does not write on the MT. Thus only erasure takes place.

912 TPB NJ A "Tape to Buffer" (17. + Ai; 220ms)

The MT of the MTH specified by N is started forward. The nearest block is to be read. A block number is read and compared with #JA, and if it is equal, the control skips the next instruction in sequence; if it is not equal, the sequence is normal. The next instruction in sequence becomes operative concurrently from this time on (approx. 17 ms from the beginning of the execution). A block of information is read and stored in the core memory. Next the MT is stopped. During the operation, the TCU and No.N MTH busy-indicators are on, i.e., for 220ms in the usual case.

If errors in the character parity and the channel parity are detected,

the TC-indicator (and the lamp on the console) is set on, but the computation does not stop.

N.B.1. The execution of the instruction is delayed until the TCU and the No.N MTH busy-indicators in the "on" state by previous instructions are restored to the "off" state.

2. If the No.N TE-indicator has been on, the operation is equivalent to NOP. If the No.N TE-indicator is set on during the read operation, the operation is still performed as usual.

3. If no block can be found, the computer searches until the tape end is detected, and the operation becomes NOP.

914 BLS NJ A "Block Search" (0.45 + Ai; 100ms + 120ms)

The MT of the MTH specified by N is started forward. A block whose block number #JA is searched for reading. Each block number is successively read and compared with #JA, and when a block number equals #JA for the first time, this block of information is read and stored in the core memory. Next the MT is stopped.

The character parity is examined for all the block numbers which are read and compared. The character parity and the channel parity are examined on the block information whose block number is #JA. If parity error is checked, the TC-indicator (and the TC-indicator lamp on the console) is set on, but the computation does not stop.

N.B.1. The execution of the instruction is delayed until the TCU and the No.N MTH busy-indicators in the "on" state by previous instructions are restored to the "off" state.

2. After 0.45 ms from the beginning of the operation, the next instruction in sequence becomes operative concurrently. During the operation, the TCU and the No.N MTH busy-indicators are on. The time needed depends upon the place of the block, and is roughly equals to (100ms)* (a number of blocks passed under the reading head) + 120ms.

3. If the TE-indicator has been on, the operation is equivalent to NOP. If the TE-indicator is set on during the reading of the object block, BLS is performed as usual.

4. If no object block can be found, the instruction searches until the TE is detected, and the operation becomes NOP. If the computer fails to detect the block, it is often useful to depress the HALT button and the TE-button on the console, then the current BLS operation becomes NOP and the computation stops.

934 TTP N. - "Test Tape" - (0.45 + Ai; 220ms)

The MT of the MTH specified by N is started forward. The nearest block is read only for checking the character parity and the channel parity without storing the information in the core memory. The MT is then stopped. If parity error is detected, the TC-indicator (and the lamp on the console) is set on, but the computation does not stop.

N.B.1. The same notes as for TPB (N.B.1,2,3) are applicable.

2. After 0.45ms from the beginning of the execution, the next instruction in sequence becomes operative concurrently. During the operation, the TCU and No.N MTH busy-indicators are on (ordinary ca. 220ms).

932 BST N. - "Back Space Tape" (0.45 + Ai; 220ms)

The MT of the MTH specified by N is started backward. The nearest block is read backward only for checking the character parity and the channel parity without storing the information in the core memory. The MTH is then stopped. If parity error is detected, the TC-indicator (and the lamp on the console) is set on, but computation does not stop.

N.B.1. The execution of the instruction is delayed until the TCU and the No.N MTH busy-indicators in the "on" state by previous instructions are restored to the "off" state.

2. After 0.45 ms from the beginning of the execution, the next instruction in sequence becomes operative concurrently. During the operation, the TCU and the No.N MTH busy-indicators are on (ordinary approx. 220ms)

3. If the MT is at load point, the operation is equivalent to NOP. During the BST operation, the load point is never detected (c.f. N.B.5 of BTP) in the usual case.

4. If no block can be found, the computer searches until the load point is detected, and the operation becomes NOP.

5. As a result of the beckward motion, the No.N TE-indicator is turned off.

930 RWD N. - "Rewind" (0.45 + Ai; 20ms; av. approx. 450cm/sec)

This instruction causes the MTH specified by N to rewind the MT to its load point.

N.B.1. The execution of this instruction is delayed until the TCU and No.N MTH busy-indicators in the "on" state by previous instructions are restored to the "off" state.

2. If the MT is at load point, the operation is equivalent to NOP.

3. As a result of the backward motion, the No.N TE-indicator is turned off.

4. After 0.45ms from the beginning of the execution the next instruction in sequence becomes operative concurrently. After approx. 20ms (= time for TCU busy) from the beginning of the excution, MTH other than No.N can be used concurrently.

5. The time needed for the rewinding of the MT (= time for No.N MTH busy) depends upon the length of the MT to be rewound. As a very rough estimate, the speed of the rewind is about three times faster on the average than the ordinary 150cm/sec.

950 JTG .J A "Jump on Tape Good" (0.45 + Ai)

The control jumps to loc.#JA if the TC-indicator is off (the MT is supposed to be good), otherwise in normal sequence. The TC-indicator is turned off at the end of the operation.

N.B.1. The execution of this instruction is delayed until the TCU busy-indicator in the "on" state by previous instructions is restored to the "off" state.

2. The time needed for the execution is 0.45ms.

952 JTE NJ A "Jump by Tape End" (0.45 + Ai)

If the TE-indicator of the MTH specified by N is on, the control jumps to #JA, otherwise in normal sequence. The TE-indicator is not affected by this instruction.

N.B.1. The execution of this instruction is delayed until the TCU busy-indicator in the "on" state by previous instructions is restored to the "off" state.

2. The time needed for the execution is 0.45 ms.

920 DMB .J A "Drum to Buffer" (2.90 + Ai + Aa)

c(E), ..., c(EL) → c(4200), ..., c(4200 + EL - E).

E = #JA, $0 \leqq EL - E < 50$, and $EL \equiv 49$ in modulo 50.

The c(E) replace the c(4200), ..., and the c(EL) replace the c(4200 + EL - E). Thus a group of words is transferred to the core memory from the drum memory (both normal and quick access bands).

N.B.1. Contents of registers other than the c(4200), ..., c(4200 + EL - E) are unchanged.

2. If $4200 \leqq E \leqq 9999$, zoros replace the c(4200), ..., c(4200 + EL - E).

3. A number in the Xx part is neglected in the operation.

4. If the core memory has been busy with some tape operations, the execution is done after the busy-indicator is off.

922 BDM .J A "Buffer to Drum" (2.90 + Ai + Aa)

c(4200), ..., c(4200 + EL - E) → c(E), ..., c(EL).

E = #JA, $0 \leqq EL - E < 50$, and $EL \equiv 49$ in modulo 50.

The c(4200) replace the c(E), ..., and the c(4200 + EL - E) replace the c(EL). Thus a group of words is transferred to the drum memory (both normal and quick access bands) from the core memory.

N.B.1. Contents of any registers other than the c(E), ..., c(EL) are unchanged.

2. If $4200 \leqq E \leqq 9999$, the operation equals NOP (514).

3. A number in the Xx part is neglected in the operation.

4. If the core memory has been busy with some tape operations, the execution is done after the busy indicator is off.

4.7 Conclusion

The details of the magnetic tape system of the KDC-I have been described with emphasis on design features and problems.

In spite of the difficulties at the time of the design of the system, the completed magnetic tape system is satisfying the required functions and is utilized almost daily at the Kyoto University Computation Center. A large scale problem was solved by using the magnetic tape system for more than one hundred hours.

Several comments obtained from the experience of designing and the operation of the system should now be added.

(1) The time displacement error is corrected to a satisfactory degree in the input circuit of the TCU; thus it did not create any trouble.

(2) The variable block number is utilized in various ways.

(3) The repertory of tape instructions is not very rich, however, it is almost enough for this type of computer.

(4) Since the MTH has an erase head and all portions of the MT are erased, high tolerance of the deviation of the relative position of the read/write head and the MT can be permitted, thus a better S/N ratio for read-out signals can be expected.

(5) In actual operating experience, parity checking has been found to be very reliable.

(6) A two-gap head in which one gap of the head is for writing and another for reading should be used for better operation. In case of this system information is written in forward direction of the MT and tested backward, and so on. Thus the operation of the system sometimes very much resembles a life test of the MT and the MTH in many cases.

(7) The block beginning and ending codes are used as an information for stopping the MT. But it would be better to use the information, "the MT is started", and "information is no longer read out", for this purpose.

(8) The number assigned to the MTH cannot be changed by the switch in this system. But is has become clear that this is more inconvenient than was foreseen.

(9) The length of the MT between the read/write head and the tape-end sensors is not constant since there is a pneumatic tape buffer between them. It would be more convenient if this length could be kept constant.

(10) The MT is stopped ca. 200ms after the tape end is detected by the use of a time-delay relay circuit. Thus it is possible to write normally if the tape end is being detected during writing operations. But it would be better if the MTH could detect these two stages of tape end in relation to a certain fixed position of the MT. This and the former items are, however, not so important since tape ends are rarely used because of a strong probability of damage during handling of the tape.

(11) A large percentage of the starting and stopping time of the MTH (7ms) consists of the operation time of the relays in the MTH. It would be possible to achieve a much higher operation speed of the MTH.

(12) Many relays are used in the MTH, some of them at a rather high current level. The micro-relays at the pneumatic tape buffers are operated very often; moreover the life of the relays of this type is not very long. most of the difficulties occurring during the operation of the system have been caused by malfunction of the relays. It is strongly recommended that an electronic circuit should replace them.

(13) It is true that almost all kinds of presently available MTHs are not satisfactory from the standpoint that the MT should not be damaged by the malfunction of the MTH, though this is rare. There are indeed many cases where an MTH handles an MT whose contents are hard to reproduce.

(14) The shielding of the read/write head and the relay circuits had to be done well beforehand. It can be said that the adjustment of the system began with the work of suppressing noise and ended with the same work.

(15) The use of suitable static circuits would have been more economical than the use of dynamic flip-flop circuits in the TCU. The main reason for this comes from the difference between the operation time needed for the control of the MTH from the operation time of the dynamic flip-flop circuit.

Chapter 5

INITIAL READ-IN ROUTINES

There are various types of computers concerning the methods of loading programs. Some computers load programs by a built-in function of the machine, while others by the aid of an initial read-in routine which has been stored in the memory in some special way. The KDC-I is that type of computer which loads programs by using an initial read-in routine. Various routines for this purpose can be considered. However, a 7-word routine has been selected as the standard routine. Input Routines of much complexity such as assemblers are read into the computer by this initial read-in routine.

Since at the very beginning an initial read-in routine can not be read by another initial read-in routine, it is read and stored manually at the operator's console. In the case of the KDC-I, this manual procedure of reading the initial read-in routine is not very simple, requiring about twenty to thirty button-pushings. Various procedures have been suggested, and at one time Professor T.Kiyono who is a head of the computation center suggested offering a prize to the one who succeeds in minimizing the steps of his procedure. The author had suggested a somewhat simpler procedure. Professor T.Kiyono then has devised a much simpler one. Assistant Professor S.Hoshino also presented a simpler procedure. Finally the author has suggested the most simple procedure and gave a proof that it is a minimum one.

The initial read-in routine, though its proceeure of loading might be tedious, is retained in the Storage Registers if it is once stored unless someone destroys it with a faulty program. Thus there might not be an important practical meaning even if one could devise a simpler procedure. Moreover the author does not fully grasp what meaning exists in a thorough investigation of this problem, but still thinks it would be important in the

studies of automata.

In this chapter, the standard initial routine is explained first. Various procedures of reading the initial read-in routine are then given. Finally the minimum procedure is presented.

5.1 A Standard Initial Read-in Routine (RDIN)

At the time of the design of the computer various initial read-in routines were considered. However, the following initial read-in routine is currently used as a standard routine;-

Instruction Location	Symbolic Instruction Function	Index	Address	Instructions in the Storage Registers
0000	RIN	00	14	633 000 0 0014
0001	STO	00	2	300 000 0 0002
0002	(any)			(any)
0003	JMP	00	0	714 000 0 0000
0004	RIN	00	14	633 000 0 0014
0005	STO	10	0	300 100 0 0000
0006	JXR	10	4	852 100 0 0004

As the name of this routine, RDIN is often used in this chapter. Since at the very beginning an initial read-in routine cannot be read by another initial read-in routine, it is read and stored manually at the operator's console. Programmers can suppose that the standard initial read-in routine is a part of the computer and has been stored in the Storage Registers (SRs) whose locations are from 0 to 6.

The instruction from which the computation starts can be decided from the contents of the Instruction Location Counter (LC). Any number can be placed in the LC by the execution of the instruction JMP at the console. If zero is placed in the LC and the computation is started, the program proceeds from there. There are various ways to utilize this routine. Since this is a readin routine, this routine reads and stores some other program.

The following situation is considered;- A program should be read and stored in the SRs whose locations start from 3, and the program should be instantly started from loc. T. The program tape should be prepared in the following way;-

(1) 0 d

(2) 830 100 w S " (SEX 10 S)

(3) 4 d

(4) The program

(5) T d

The computation of the initial read-in routine begins from loc.0, or by the instruction RIN/, since 0d is read, the control jumps to loc.0 in any case.

(i) By the instruction in loc.0, the instruction "SEX 10 S" is read into the UA.

(ii) By the instruction in loc.1, this instruction in UA is stored in the loc.2.

(iii) By this newly stored instruction "SEX 10 S", S is placed in the IR 1.

(iv) By the instruction in loc.3, control jumps to loc.0 again.

(v) By the instruction in loc.0, 4d is read, and control jumps to loc.4.

(vi) By the instruction in loc.4, the first word of the program is read into the UA.

(vii) By the instruction in loc.5, this word in UA is then stored in the location specified by the c(IR1).

(viii) By the instruction in loc.6, the c(IR1) is increased by one and the control jumps to loc.4 once again.

(ix) This process is repeated until all of the words in the program are read and stored in consecutive locations.

(x) By the instruction in loc.4, Td is read and control jumps to loc.T.

(xi) The program then starts from there instantly.

From the above discussion, the first part of the initial read-in routine in loc.0-3 is a routine which performs the computation of instructions on paper tape. The second part of the routine is thus the core of the read-in routine by the aid of the IR 1. Since the instruction RIN can read an instruction, a fixed or a floating point number by the aid of the special characters or the TCC, the core of this routine requires only three instructions, which can be considered one of the simplest routines of this kind.

5.2 Various Methods of Reading the Routine RDIN

Before describing various methods, it is necessary to describe those functionsof the operator's console which have a relation to this problem. Any instruction of the computer can be set and executed manually at the operator's console. An instruction word is prepared by setting the REGISTER SET buttons (or switches) in the middle of the panel of the console. But before setting an instruction in the OR (Order Register), the computer must be in the state of HALT (confirm that the HALT lamp is on). By depressing the OR CLEAR button and next depressing the OR SET button, the instruction can be set in the OR. If the SS button at the lower right of the panel is then depressed, the execution of the instruction takes place.

If the START button is depressed, the computer takes the next instruction from the Storage Register (SR) whose location is specified by the c(LC), and proceeds from there.

Therefore it is possible to read any instruction to the UA by the manual execution of the instruction "RIN/". It is also possible to store the c(UA) at any position in the SRs by the manual execution of the instruction "STO".

1. A Primitive Method

Since any instruction can be read and stored in any position in the SRs by the manual execution of the proper instructions, each instruction in the standard initial read-in routine RDIN can be read and stored in each corresponding location by repeating the above process. Thus the details of this method is as follows;-

The necessary program tape

The following program tape should be prepared and placed in the input unit.

(a)	633 000 w 14 "	(*RIN/ 00 14)
(b)	300 000 w 2 "	(STO 00 2)
(c)	714 000 w 0 "	(JMP 00 0)

```
(d)  300 100 w 0 "   ( STO 10  0 )
(e)  852 100 w 4 "   ( JXR 10  4 )
```

Next the following instructions should be executed manually at the console. Instructions are written using the symbolic codes.

The manual operations and remarks

```
(1)   RIN/ 00 14     c(UA) = (a)
(2)   STO  00  0     c(0) = (a)
(3)   STO  00  4     c(4) = (a)
(4)   RIN/ 00 14
(5)   STO  00  1     c(1) = (b)
(6)   RIN/ 00 14
(7)   STO  00  3     c(3) = (c)
(8)   RIN/ 00 14
(9)   STO  00  5     c(5) = (d)
(10)  RIN/ 00 14
(11)  STO  00  6     c(6) = (e)
```

This method requires therefore eleven steps of manual execution of instructions. Thus this method is obviously tedious. To reduce the trouble, the following method had been suggested and has been used at the Computation Center for some time.

2. The HH Method

Since this method was suggested by Professor H.Hagiwara, the above name is given to this method.

The necessary program tape and the manual operations required for the loading of the routine RDIN are as follows. Remarks are also given for convenience in understanding the operations.

The necessary program tape

```
(a)  633 000 w 14 "   ( RIN/ 00 14 )
(b)  300 010 w  0 "   ( STO  01  0 )
(c)  852 100 w  4 "   ( JXR  10  4 )
(d)  714 000 w  4 "   ( JMP  00  4 )
(e)  830 100 w  0 "   ( SEX  10  0 )
(f)  714 000 w  4 "   ( JMP  00  4 )
(g)  8d               ( 8d )
(h)  633 000 w 14 "   ( RIN/ 00 14 )
(i)  300 000 w  2 "   ( STO  00  2 )
(j)  710 000 w  0 "   ( HJM  00  0 )
(k)  714 000 w  0 "   ( JMP  00  0 )
(l)  2d               ( 2d )
```

The manual operations and remarks

(1)	SEX 10 6	c(IR1) = 6
(2)	RIN/ 00 14	c(UA) = (a)
(3)	STO 00 4	c(4) = (a)
(4)	RIN/ 00 14	c(UA) = (b)
(5)	STO 00 5	c(5) = (b)
(6)	JMP 00 4	jump to loc.4 and the computer takes control and proceeds from loc.4.

At the end of the step (5), the contents of the drum are;- c(4) = RIN/ 00 14 , and c(5) = STO 01 0. Furthermore the c(IR1) = 6. Afterwards the c(LC) is set to 4 and the computer takes control and proceeds from there. By carefully tracing the steps, it can be understood that the routine RDIN is stored in the drum.

This method has greatly simplified the former primitive method. However, care should be taken that at the end the c(7), c(8) and c(9) are replaced by (JMP 00 4), (SEX 10 0) and (JMP 00 4) respectively.

3. The TK method

This method was suggested by rof. T.Kiyono, who has named this method a YK method in which Y is taken from the author's name, in spite of my very little contribution to this method . Thus the author should be allowed to call this method the TK method. The manual operations, the necessary program tape and remarks are as follows;-

The manual operations and remarks

```
(1)   RIN/ 00 10     read (a)
(2)   STO  00  0     c(0) = RIN/ 00 14
(3)   STO  00  3     c(3) = RIN/ 00 14
(4)   RIN/ 00 11     read (b)
(5)   STO  00  1     c(1) = STO  00  2
(6)   RIN/ 00  1     read d
```

The necessary program tape and remarks

```
(a)        RIN/ 00 14     stored in loc.0 and 3.
(b)        STO  00  2     stored in loc.1.
(c)   d
(d)        SEX  10  1     c(Ind) = 1
(e)   0d   LDN  00  0     c(MD) = RIN/ 00 14
(f)   0d   STM  00  4     c(4) =  "
(g)   0d   STM  00  6     c(6) =  "
(h)   0d   PAM            c(MD) = PAM
(i)   0d   STM  00  5     c(5) =  "
(j)   4d   JXR  10  4     saved in MD.
(k)   0d   STM  00  6     c(6) = JXR  10  4
(l)   4d   JMP  00  0
(m)   0d   STM  00  3     c(3) = JMP  00  0
(n)   4d   STO  10  0
(o)   0d   STM  00  5     c(5) = STO  01  0
(p)        HJM  00  0     c(2) = HJM  00  0
```

At the end the RDIN is stored in loc.0-6 and all other registers are left unchanged.

4. Methods Which Require Some Special Conditions

If the CLOCK button on the panel of the console is reset an set again before applying the methods, it can be expected that the c(LC) = 0, c(IRs) = 0 and so on. It is possible to simplify the manual procedures required in applying the methods by utilizing these conditions. One example, suggested by the author, is as follows:

The necessary conditions c(LC) = 0 and c(IR1 & 2) = 0.

The manual operations and remarks

```
(1)  RIN/ 00 20     read (a)
(2)  STO  00  0     c(0) = RIN/ 00 14
(3)  STO  00  2     c(2) = "
(-)  SS             depress SS button; execute c(0) & jump to loc.2.
(4)  STO  00  1     c(1) = STO  20  2
(-)  SS             depress SS button; execute c(2) & jump to loc.0.
(5)  STO  00  3     c(3) = JMP 00  0
```

The necessary program tape and remarks

```
(a)  RIN/ 000 w 20 "      read by (1)
(b)  STO  200 0 0002d     read by c(0) and jump to loc.2.
(c)  JMP  000 0 0000d     read by c(2) and jump to loc.0.
(d)  JXR  200 w  0 "
(e)  JMP  000 w  0 "
(f)  RIN/ 000 w 20 "      read by Bootstrap in loc.0-3 by IR2.
(g)  STO  100 w  0 "
(h)  JXR  100 w  4 "
(i)  JMP  000 w  0 "
(j)  4d
(k)  RIN/ 000 w 20 "      read by Bootstrap in loc.4-7 by IR1
(l)  STO  000 w  2 "
(m)  HJM  000 w  0 "
(n)  2d                   jump to loc.2, or HJM.
```

At the end the RDIN is stored in loc.0-6, and in loc.7 (JMP 00 4) is stored.

This method requires a somewhat simpler manual procedures than that of the TK method, though some special conditions are needed. The idea of this method lies in the steps (b) and (c) in which instructions and the code d are suitably combined.

The Modified TK Method

In the case of the TK method, Prof. T.Kiyono suggested that it was also possible to reduce the manual procedure if the condition c(LC) = 0 were fulfilled. Moreover the same program tape can be used.

The necessary condition c(LC) = 0.

The manual procedures

```
(1)   RIN/ 00 10
(2)   STO  00  0
(3)   STO  00  3
(-)   SS           depress SS button; excute c(0).
(4)   STO  00  1   (START).
```

Since it is not too difficult to trace how the routine RDIN is produced, an explanation is omitted.

5. The SH Method

Assistant Professor S.Hoshino suggested this method. The method does not require any special conditions and requires only four steps of manual operations.

The manual operations

```
(1)   RIN/ 00 13
(2)   STO  00  3
(3)   STO  00  4
(4)   RIN/ 00 14
```

The necessary program tape

```
(a)   STO  400 w  1 "
(b)   RIN/ 440 0 0004 d
(c)   SEX  100 0 0003 d
(d)   STO  100 0 0003 d
(e)   JXL  100 0 0004 d
(f)   LWX  140 0 0003 d
(g)   STO  140 0 0003 d
(h)   JXL  100 w "
(i)   STO  100 w 13 "
(j)   RIN/ 000 w 14 " d
(k)   JXR  100 w  4 "
(l)   STO  100 w "
(m)   RIN/ 000 w 14 "
(n)   JMP  000 w "
(o)   SEX  100 w 1 " 4d
(p)   STO  000 w 2 " d
(q)   HJM  000 w "
```

At the end the RDIN is stored in loc.0-6, all other registers being left unchanged. The main features are the usage of RIN/ 440 0004d, and so on.

6. The SY Method

Finally the author suggested this method. Untill now many methods have been suggested and most efforts are exerted in simplifying the manual procedures used in each of their methods. The manual operations require 11 steps at the beginning, next 6 steps and finally 4 steps in the SH method. The author's method requires 3 steps, moreover the author succeeded in proving that this is the minimum procedure. By this way the problem of simplification has finally been settled.

The manual operations

(1)	RIN/ 00 10	
(2)	STO 00 0	
(-)	SS	depress the SS (Single Step) button.
(3)	RIN/ 00 10	depress the START button.

The necessary program tape

```
(a)  STO  400 w  1 d          loc.0  STO  40  1
(b)  RIN/ 000 w 14 "          loc.1  STO  40  1
(c)  LDM  000 w  2 d          loc.2  RIN/ 00 14
(d)  STM  000 w 4 d           loc.4  RIN/ 00 14
(e)  PAM  000 w d
(f)  STM  000 w 3 d           loc.3  PAM
(g)  STO  100 w 9999 "
(h)  STM  000 w 5 d           loc.5  STO 10 9999
(i)  JXR  100 w  4 "
(j)  STM  000 w  6 d          loc.6  JXR 10    4
(k)  SEX  100 w 1 d           by using the program in loc.4-6, the
(l)  4 d                      rest of the tape is to be read and stored
(m)  RIN/ 000 w 14 "          into the loc.0-5.
(n)  STO  000 w 2 "
(o)  HJM  000 w "
(p)  JMP  000 w "
(q)  RIN/ 000 w 14 "
(r)  STO  010 w "
(s)  2 d
```

At the end the RDIN is stored in loc.0-6, and all other registers are left unchanged. No special conditions are required. The main features are the usage of the codes w and d, etc. These will be fully discussed in the proof.

5.3 The Minimum Procedure of Reading the Routine RDIN

The purpose of this section is to give a proof that the SY method is the sole minimum procedure (uniqueness) among various procedures of reading various types of KDC-I initial read-in routines (abbr. RN) (generality), including the standard routine (abbr. RDIN) now utilized, and at the same time to present a minimum step RN.

The definition of the minimum procedure of reading the RN which is employed here and considered appropriate is to produce the RN on the Storage Registers (whose locations are from loc. 0 to 6 in the case of RDIN) with a minimum number of manual button-pushings at the computer console under the assumption that all buttons are in reset conditions at the beginning except the POWER, DRUM, CLOCK, TIMER, and PTR Select, and that the computer is at the HALT stage. At this time the complexity of the paper tape programming is not taken into account. The number of button-pushings for the SY Method is 7 + 7 + 1 = 22, which is a minimum.

The proof will be given according to the following procedure:

(1) The machine characteristics relevant to this problem are explained. (Paragraph 1)

(2) The manual procedure is then formalized.(Paragraph 2)

(3) Next some theorems are set up. (Paragraph 3)

(4) Starting from the simplest form of the manual procedure, more complex ones are investigated step by step until a minimum procedure of reading the RN is reached.(Paragraph 4)

(5) A minimum step RN is shown. (Paragraph 5)

(6) The uniqueness and the generality of the solution is proven.(Paragraph 6)

1. Relevant Machine Characteristics

(1) RIN The characteristics of the tape control code could be utilized.

For example :

(BY RIN/) Ad : A is read to UA and control jumps to loc.A.

wAd : A is read to UA and control jumps to loc.0, not loc. A.

w: 5 digit left cyclic shift at UA and next character A is read to [illegible] temporarily.

d: jump to 0 = c(UA)ad & A is added to UA from [illegible] & sense the end for RIN/.

(BY RIN/) 300 000 0 0006d is read & jump to loc. 6.
300 000 w6d is read & jump to loc. 0.
3ww6d is read & jump to loc. 0.
-----" is read & normal sequence.
} & all will result in the same c(UA).

(2) STO 4 1 If this is in UA and if stored and executed repeatedly, all SR are filled with this information.

(N.B.) Some combination of STO 4 1, STM 4 1 and SLA 4 1 shows also some such properties.

(3) PAN PAN in UA, if stored and executed, will treble itself in MD, SR and original UA, whereas an ordinary instruction does or does not double, or vanishes by the same procedure.

(N.B.) It might be said that the degree of amplification of information is 3 in PAN, 2 or 1 or 0 in the ordinary case etc. Furthermore, if allowed, the generative power of STO 4 1 could be said to be infinite, etc.

(4) Compound instructions They are sometimes very useful.

2. The manual Procedures

The Manual Procedures are described by utilizing the following symbols:-

M = (REGISTER SET, OR CLEAR, OR SET and SS are executed at the console.) I.e., an execution of a manually set instruction.

a = (SS is executed.) I.e., an execution of a stored instruction.

s = (START is executed.) I.e., an execution of a stored program.

(N.B.1.) All manual procedures could be described as a sequence of the above steps, e.g. Ma, MMaMs, etc.

(N.B.2.) The sequence "as" is equal to "s".

(N.B.3.) aa a, aa....a a, etc.

3. Some Theorems

Theorem 3.1 It is necessary for RN to contain explicitly or implicitly a set of orders which makes possible the transfer of information from the input device to the storage register (SR).

e.g. RN must contain; RIN & STO, (a minimum set of orders)

or RIN & PAM & STM,

or RIN & Shift & SLA (FSL),

etc.

Proof: Self-evident.

(N.B.) "Implicitly" means that a program which can generate an order such as RIN but does not contain RIN explicitly, still can be an RN.

(N.B.) There is indeed a case that only an instruction STO in loc.0 can generate RN if STM is in UA and RIN is in MD.

Next, some other theorems are set up through the investigation of the possibility of producing an RN by only one instruction in the SR.

However, the following conditions are assured:-

1) An instruction X is in loc. A. ($0 \leqq A \leqq 6$ in case of RDIN)

2) Any other registers and paper tape exept SR and OR contain necessary

information.

3) At the beginning, the START button will be pushed, which means c(LC) = A is assumed.

(3.1) The property of X in loc. A

Since X is a single instruction in the SR and automatic computation must be done;-

The property of X must be { do (something) & jump to loc. A (Self)...(i) case
" A+1(Next)...(ii)
" B (≠A or A+1) ...(iii)

Theorem 3. 2. X cannot be a "do & jump to self" instruction. (case (i)).

Proof: Jump orders (JMP, FFX, Non-div, JXL, etc.) do nothing. RIN(& jump by d) only changes the c(UA). QED.

From case (iii) follows that an instruction must be stored in B and jump to B (or self). But such X does not exist.

Theorem 3. 3. There is no instruction which stores and jumps.

Theorem 3. 4. The X, which is a single instruction in the SR, must be at least a "store next" instruction,

i.e. STO, STM, or SLA (FSL) Next.

Proof: Cases(i) and (iii) do not hold, and the meaning of case (ii) is that some instruction must be stored in loc. A+1 and the program sequence must be normal. QED.

By Theorem 3.4., an instruction, say Y, is stored in loc. A+1 by X.

(3. 2) The property of Y in loc. A+1

The property of Y must be do (something) & jump to loc. A ...(i)

" A + 1 ...(ii)

" A + 2 ...(iii)

" B(≠A, +1, +2) ...(iv)

Case (iv) does not hold by Theorem 3. 3.

Case (i) and (ii) are also impossible since jump orders do nothing at all, and there is no "store A + 2 and jump to A, A + 1 or B" order. Even Y = RIN will be impossible.

Theorem 3. 5. STO Next and RIN in consecutive locations can not produce RN.

i.e. loc. A STO Next

loc. A + 1 RIN

Proof: If RIN is effectively a "read and jump to A + 1", refer to Theorem 3. 2.

If RIN is a read and jump to A, the same procedure is only repeated. QED.

For the case (iii), the same situation exists as for X.

Theorem 3. 6. The X in loc. A and the Y in loc. A + 1 must be both "store next" instructions.

By this, an instruction, say Z is stored in loc. A + 2 by Y.

(3. 3) The property of Z in loc. A + 2

The property of Z will be outlined as follows:

do(something) and jump to loc. { A, A + 1, A + 2 } ... i)

A + 3 ... Z must be again store next,...(ii)

B (=A, ..., A + 3) --- Impossible

Theorem 3. 7. Z could be either RIN (i) or store next (ii).

Proof: In (i); No "store and jump" orders, jump orders do not accomplish anything. Case (ii) is clear. QED.

Theorem 3. 8. It is known that the following can be an RN (Z is RIN).

loc. 0 STO Next

loc. 1 ----

loc. 2 IN one word (start from loc. 2.)

It can be said, in other words, that this is a minimum form of RN.

Proof: The above is the core of the SY method. And indeed, the above situation exists.

e.g. The first condition c(0) = STO Next, c(UA) = STO Next and c(RD) = RIN one word. QED.

If Z is a "store next" instruction and by it, say, U is stored in loc. A + 3 by Z, in exactly the same way U must be either RIN or a store next instruction. This fact is consistent with Theorem 3. 1.

(3. 4) RIN and STO in the SR

The following situation is assumed : there are RIN and STO in the SR, the same STO in UA, and no other information is available in the computer, and the execution by START begins from RIN or STO.

Theorem 3. 9. The following cannot be an RN.

loc. A RIN

A + n STO ($n \geq 1$)

Proof: Any instruction next to STO must be a jump type. Jump orders do not accomplish anything at this stage. "Read-in and Jump" also accomplishes nothing for RN. QED.

Theorem 3. 10. The following cannot be an RN.

loc. A STO

A + n RIN (n ≥ 4)

Proof: Whether one begins with STO or RIN (after RIN, STO must be executed), STO must be STO next type (or X is read and stored in loc. A + 1 and will be executed). X cannot be a store next, but must in any case be a jump type. If X is an ordinary jump type, it accomplishes nothing. If X is RIN & d jump, the next step must be taken from loc. A or A + 1, or A + n, all of them having no effect. QED.

4. The Minimum Procedure

Starting from the simplest form of the manual procedure, more complex ones are investigated step by step until a minimum procedure of reading the RN is reached.

(4. 1) Ms: since no instruction can be stored in SR, no possibility for any RN.

Ma M ... etc. : a should not be used.

(4. 2) MMs: One instruction can be stored by MM, but no possibility for any RN by Theorem 3. 4. (an instruction in SR must be STO Next) and Theorem 3. 1. (... in any case RIN is necessary).

(4. 3) MMMs: The consideration of the following two cases is necessary and sufficient.

(case 1) Three instructions of the same type can be in the computer (by MMM).

i.e. One in UA, the other two in SR, or each in UA, SR and in MD or LA respectively.

(MMMs = RIN. STO. STO. s. etc.)

(case 2) One instruction in SR and a new instruction in UA.

(MMI s = RIN. STO. RIN. s.)

Again no possibility is expected for any RN.

Proof: Three instructions cannot be stored by MM.

(case 1) Any two steps of the same instruction cannot produce both RIN & STO. (refer to Theorem 3. 1.)

(case 2) Refer to:-

Theorem 3. 4.(a single instruction in SR must be STO Next),

Theorem 3. 1.(...c(UA) must be RIN) and

Theorem 3. 5.(STO Next & RIN in consecutive locations will never produce any RN). QED.

(4. 4) MMaMs: Only the following cases should be examined:-

(case 1) RIN. STO. a^p. RIN. s.

(case 2) RIN. STO. a^p. STO. s.

($p \geq 1$)

Proof: MMa must be RIN. STO. a., because of a.

If an instruction other than RIN or STO is read, stored and executed by MMa, it is not possible to have both RIN and STO in SR by the third M explicitly or implicitly (Theorem 3.1.).

If STO is read, stored and executed by MMa, then RIN must be read by the third M (case 1).

If RIN, STO must be read and this must be stored by the third M (case 2). QED.

Theorem 4. 1. RIN. STO. a. RIN. s. can be a procedure for RN, and a sole minimum procedure of this type is the SY method.

Proof: STO Next should be read and stored and executed by manual RIN. STO. a. So the form of the STO Next must be ;-

(i) 300 400 w 1d (if read, will jump to loc. 0. so in this case it must be stored manually in loc. 0.)

(ii) 300 400 0 Ad (it must be stored in loc. A and also must be STO Next, so 2A = A + 1 or A = 1)

(iii) 300 440 0 Ad (impossible, since 3A ≠ A + 1)

From case (i) the SY method is derived,

i.e., RIN/10 . STO 0 . a . RIN/10 . s . (22 button-pushings).

It cannot be RIN/9 or less, since it must read somehow RIN/ 000 w 14 " (10 characters). If it is RIN/ 11 or 14, etc.. it is also possible to be a procedure for an RN, but the number of button-pushings will become 8 + 8 + 1 + 8 = 25, which exceeds 22.

The case (ii) will not be fully examined if it could produce any RN, but apparently it is less simple than the SY method, since it must be RIN/ 12 . STO 1 . a . RIN/ 11 . s . etc. (8 + 8 + 1 + 7 = 24 button-pushings).

QED.

Corollary: The SY method would be simpler than RIN. STO. a . RIN. s., even if these cases arise.

Proof: The above proof can be applied; so if these cases would arise, the number of button-pushings would exceed that of the above by at least one. QED.

(4. 5) The case "MMaMs = RIN. STO. a. STO. s".

By Theorem 3. 1., RIN is read, stored and utilized (RIN & jump to self) and STO is read and stored (STO is also in UA). So paper tape begins with RIN, STO, ...

So the following five cases must be examined.

	loc.			
(1)	0	RIN one word.		
(2)	1	RIN 44 1	:	RIN 3 characters max.
(3)	2	" 2	:	" 6 "
(4)	3	" 3	:	" 9 "
(5)	A(≥ 4)	" A	:	" 3A or one word.

(1) Not possible by Theorem 3.9

(2) Not possible since RIN 3 characters can only read STO 0 (3ww) and "STO 0 & RIN 3" has no effect.

(3) It can read STO as 3wwAd which jumps to zero.

And STO must be STO next type, so A = 1.

i.e. loc. 0 STO 00 1

1 -----

2 RIN 44 2

Since RIN 4 2 reads instructions as IwwAd form, ADD(100), FAD(200), STO(300) and EAD(500) could be read, but EAD(500) cannot be used (MD cannot be referred to.) And RIN 4 2 must be RIN/ 44 2, since w is a five digit cyclic shift. If RIN 44 2 is used, manual RIN. STO. will leave the c(loc. 2) and the c(UA) as $?_1$ 632 44$?_2$ $?_3$0002 ($?_i$ is arbitrary). Even if loc. 2 is executed by manual a and if paper tape is prepared in any way possible, it is not possible to read as a result ?300 00? ? 0001 in UA. (e.g. $?_2$ = 3, $?_3$ = 0, and w" is read, it becomes 430 0 000 2 632 4.)

Because RIN/ must be used, ADD & FAD orders become meaningless (they will be executed at loc. 1).

Now it has become clear it is not possible.

(c.f.) If w were not a cyclic shift, but a short left shift, it would be possible for this case to be a RIN.

e.g. Under the following case :

Paper tape 632 448 6 0002d 3ww1d 3ww6d

1ww2d w6d 830 100 w6d

1ww2d w6d 830 200 w9999d etc.

(4) STO must be read by RIN 9 characters, STO must be read as :

(i) 3wwAd (and jump to loc. 0) by RIN/.

(ii) 0000Ad (and jump to loc. A) by 632 44 3 0 0003 and its result is

4 300 003 0 000A. Since jump to A, STO A must be stored in loc. A.

Then it is clearly not possible.

From case (i) follows : loc. 0 STO A

1 ---

2 ---

3 RIN/ 44 3

A must be 1, then the program stops.

So case (4) is impossible.

(5) The RIN in loc. A ($\geq$ 4) can read one full word, and it must read STO (Theorem 3. 1.). But this STO must be stored manually at least in loc. A-1, A-2 or A-3. (Theorem 3. 9. and 3. 10.)

From this, STO on tape cannot be 300 000 wBd (jumps to 0 and must be stored in loc. 0), or 300 .....". So it must contain d but not w. And clearly it can't be 300 000 0 A d. Then after the execution of RIN. STO, a. STO, newly stored STO in loc A-1, A-2 or A-3 must be executed. If 300 000 0 A-1(or A-2 or A-3) d, which must be stored in A-1(or A-2 or A-3 respectively), or STO Self is read and utilized, the program stops in case of STO A-2 or A-3, and even STO A-1 is not useful. (Since next step RIN might read some specific instruction, and it might be stored in loc. A-1, but STO type instruction already in loc. A-1 should not be destroyed, then it must again the same STO A-1.)

So only possibility now is either 300 400 0 A-1(or A-2 or A-3)d

or 300 440 0 A-1(")d.

The following three cases for each should be studied;-

(i) STO in loc. A-1 (ii) STO in loc. A-2 (iii) STO in loc. A-3.

In each case the effective address of STO is 2(A-1) or 3(A-1) and etc.

(i) loc. A-1 STO 2(A-1) or 3(A-1) , (A $\geq$ 4)

A RIN one word

After the execution of RIN and STO, RIN will read an instruction (X) and jump to loc. 2(A-1) or 3(A-1). (Jump to A or A-1 is nonsense.)

X must be a "do and jump" instruction, which does not contribute for RN. (even if it is a RIN and jump by d). So the case (i) is impossible.

(ii) loc. A-2 STO 2(A-2) or 3(A-2) $(A \geq 4)$

A-1 ---

A RIN one word

After the execution of STO, the program stops, since 2(A-2) can't be A-1 or $A \neq 3$(refer to case (4)), and since 3(A-2) can't be A-1 or $A \neq 5/2$.

(iii) loc. A-3 STO 2(A-3) or 3(A-3)

A-2 ---

A-1 ---

A RIN one word

STO must be STO next.

So $2(A-3) = A-2$, then $A = 4$,

but $3(A-3)$ A-2, or $A \neq \frac{7}{2}$.

Then it becomes;-

loc. 1 STO 4 1

2 ---

3 ---

4 RIN 44 4

But since program begins from loc. 1 and STO 4 1 will regenerate itself (Section 1.(2).), then it is <u>impossible</u>.

From the above discussion follows;-

<u>Theorem 4. 2.</u> Manual procedure RIN. STO. a. STO. s. will not produce any type of RN (, whose number of button-pushings might have eventually become as small as 7+7+1+4 = 19).

(4. 6.) The case $MMa^{p}Ms$ = RIN. STO. a^{p}. STO. s. ($p \geq 2$)

Again RIN must be read and executed p-times by manual a . and, by this, STO must be read. So the situation differs, if any, from that of 4. 5. by the case in which RIN can't be RIN/ (If RIN/, the situation becomes the same as 4. 5.) With referrence to 4. 5., the following cases should be examined:-

	loc.							
(1)	1	RIN	44	1	p-times	: RIN	3p	characters max.
(2)	2	"		2	"	: "	6p	"
(3)	3	"		3	"	: "	9p	"

(N.B.) Both loc. 0 case and loc. A(≥ 4) case are the same as 4. 5.

(N.B.) With RIN(not RIN/), w must be carefully used.

(1) It is not possible to read STO instruction by RIN3 twice or more. (because w can't be used from the first time.)

(2) Less than 6 characters are read by each manual a, and must jump to self i.e.,to loc. 2., but it is not possible since 300 002d can't be read.

(3) STO A could be read under the tape form such as e.g., 300 0003 dd000Ad ($p = 3$) and program step jumps to A. So it must be stored in loc. A. Next, store self operates. After reading several digits, w could be used. With w the control jumps to zero, which still accomplishes nothing.Then no possibility can be envisaged. So together with Theorem 4. 2.;-

Theorem 4. 3. Manual procedure RIN. STO. a^{p}. STO. s ($p \geq 1$) will not produce any type of RN.

Theorem 4. 4. In all three or less than three M cases, RIN. STO. a. RIN. s. is a sole solution for RN.

(4. 7.) MMMs.

Theorem 4. 5. RIN. STO. a. RIN. s. is simpler than any other

possible procedures such as MMMs, thus is a sole minimum procedure for RN.

Proof: The simplest form of the four M case is, if possible, RIN. STO. STO. STO. s. (RIN is necessary, if STO is not used, at least three kinds of instructions are necessary in four M's.), whose number of button-pushings might become as small as 7 + 7 + 4 + 4 = 22, the same as of RIN. STO. a. RIN. s. But this form cannot read any RN, since three instructions but of the same type which will be stored under this case cannot produce both RIN and STO in SR (refer to Theorem 3. 1.) and other four M case needs more button-pushings.

So from theorem 4. 4. it is now clear.

5. A minimum Step RN

From Theorem 3. 1. and 3. 8., a minimum step RN is as follows;-

(The uniqueness is not yet examined, but probably a sole one.)

```
loc. 0 STO   Next
       1 ---
       2 RIN one word  (start from loc. 2)
```

6. The Uniqueness and the Generality of the Solution

From Theorem 4. 1. and 4. 5., it is clear that the SY method is a sole minimum procedure (uniqueness) among various procedures of reading various types of KDC-I initial read-in routine RN (generality), including the standard routine RDIN.

But it must be said again, the complexity of the paper tape programming is not taken into account. Much simpler paper tape programming for the SY method than that suggested might exist. But so far it is not yet examined.

It is interesting to note that the minimum procedure read the minimum step RN of paragraph 5., and by this RN, the RDIN is read.

Chapter 6

LOGICAL DESIGN OF THE COMPUTER

6.1 Introduction

The performance of an arithmetic and a control unit depends chiefly on the performance of each basic logic circuit of the units and the method of logical design which depends largely on the logical properties of the basic logic circuit. Boolean algebra provides a mathematical framework for describing the logical design of the computer. It is however only applicable to tiny well-defined portions of a computer, e.g., such as adders, counters, etc., in most practical cases. Most theories concerning computing machineries treat least requirements for them, e.g., theories on the Turing machine are typical examples. Computers actually built are much more complex and useful ones in which convenience is an important factor of the design. Thus the theory of logical desing for a whole computer has hardly been developed. The design of the KDC-I was conducted as described in the previous chapters.

In this chapter the logical properties of the basic logic circuit of the computer is described first, then the algebraic methods of logical design are reviewed. The simplification of a Boolean function is performed by using the Veitch diagram simplification method for obtaining the simplest normal form. The author devised a method of obtaining the simplest normal form with a restriction on the operation of an input gate in which no more than a single AND gate can send binary 1 at the same time, which was for the time being necessary to cover a defect of the electrical properties of the basic logic circuit. Next, the main features of the logical design is given, and some arithmetic elements are presented to show the example of the simplest normal function with the gate restriction.

6.2 Logical Properties of the Basic Logic Circuit

The whole logic circuit of the computer is composed of many standardized basic logic circuits. The basic logic circuit of the MDC-I is a diode logic and transistorized dynamic circuit whose details will be given in chapter 7. In this section only its logical properties are presented.

The logical properties of the basic logic circuit of the computer are of the delay memory element or the D flip-flop with input gates which perform logical operations (Ph 1). The D flip-flop has a single input and an output (both "ON" and "NOT" or negated output) equal to the input one bit-time earlier. The truth table for this memory element showing its state at time $(n + 1)$, (Q^{n+1}), as a function of its input line at time n, (D^n), is given:

D^n	Q^{n+1}
0	0
1	1

The corresponding characteristic equation for the flip-flop is, of course,

$$Q^{n+1} = D^n.$$

Since the application equation is

$$Q^{n+1} = (g_1 Q + g_2 \bar{Q})^n,$$

where g_1 and g_2 represent Boolean functions of whatever variables determine the state of Q, although neither g_1 nor g_2 contains Q itself.

Then we can immediately write D in terms of g_1, g_2, and Q:

$$D = g_1 Q + g_2 \bar{Q}.$$

If $g_1 = g_2$, then $D = g$.

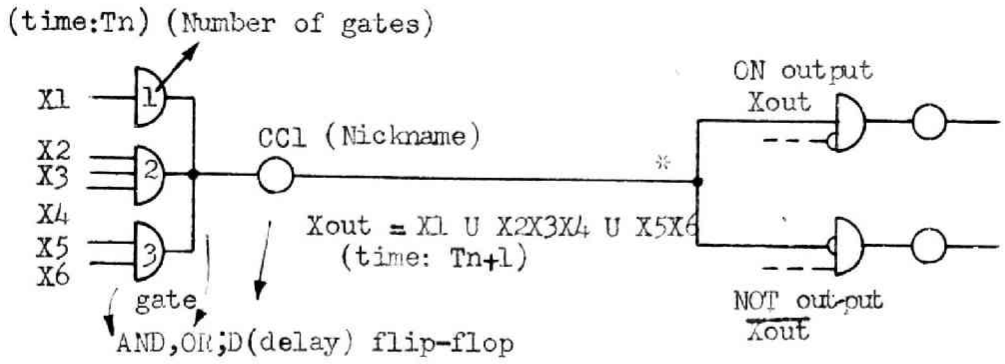
To perform the function of the application equation, diode-logic circuits which perform "AND-OR" pyramid or second-order logic are attached to the input of the D flip-flop. Thus it can perform the operation of sum of products of logical variables. In actual electric circuits,

however, limits are placed on the available number of variables both for the product and sum (fan-in limit). There are also limits on the side of the "ON" and "NOT" outputs (fan-out limit), Not only the available number of them, but also the length of them is restricted. The latter limit was not considered in the case of the logical design of the computer, it was considered at the time of allocating the positions of circuit packages. Logical diagram symbols of the basic logic circuit and its packages are
.1 shown in Fig. 6.1 together with the values of Fan-in and Fan-out limits. Electrical properties of this circuit in connection with the logical properties are given in chapter 7. By connecting pins of the A and D type packages properly, it is possible to construct a logic circuit of various complexity. The logical design of the computer was conducted by using these five kinds of packages.

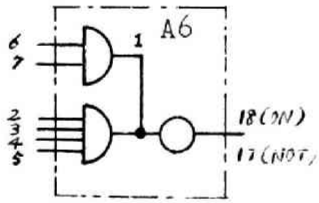
In addition to the basic logic circuit described above, a buffer amplifier which is logically an invertor without a time delay was developed. It has an input which accepts either ON or NOT output signal (a basic logic circuit can drive ten times as many inputs of buffer amplifier as that of the basic logic circuit.), and has two equivalent outputs, each of which
.2 can drive no greater than ten inputs of basic logic circuits. In Fig. 6.2 the logical diagram symbols of the buffer amplifier are shown. Since this circuit was developed when the logical design by using only basic logic circuits was roughly completed and moreover it had not enough proven characteristics, this circuit was not extensively used. However, in a timing logic circuit or in a circuit whose NOT output should be transmitted beyond the limit of the permissible wire length, the buffer amplifiers were effectively used.

6.3 Algebraic Methods of Logical Design

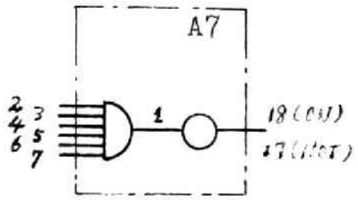
As is well-known that Boolean algebra provides a mathematical



(* The output has actually two separate lines, however, the above convention is used.)

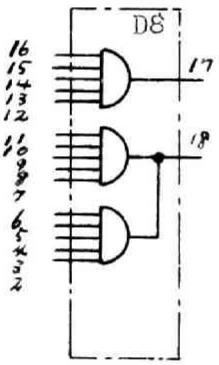


Type A6 package

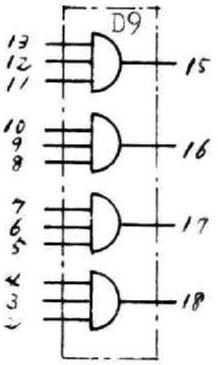


Type A7 package

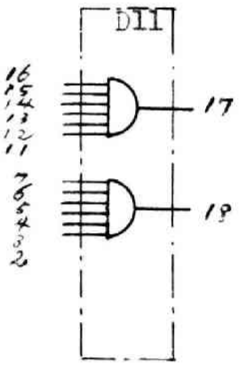
(Both ON and NOT outputs are connected to AND-gates.)



Type D8 package



Type D9 package



Type D11 package

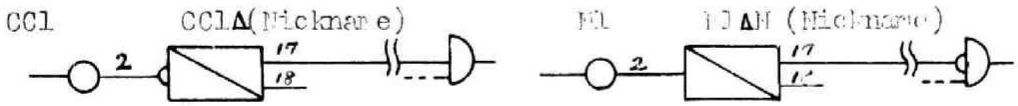
(The output of Type D package is connected to a OR-gate or No.1 pin of Type A package.)

(Both A and D type packages are of 18-pin 90- by 110-m printed card with metal frame. Pin numbers are written in the diagrams.)

Fan-in limit : 6 inputs to an AND-gate; 8 AND-gate outputs to an OR-gate; 1 OR-gate output to a D flip-flop input.

Fan-out limit: $No + Nn \leq 13$, $6 \geq Nn \geq 0$, and $No \geq 0$, where No is a number of ON outputs and Nn a number of NOT outputs.

Fig.6.1-Logical properties of the basic logic circuit of the MDC-I and its logical diagram symbols.



The NOT output of the D flip-flop is connected to the input of the invertor; thus its output is logically equal to the ON output.

The ON output of the D flip-flop is connected to the input of the invertor; thus its output is logically equal to the NOT output.

Fig.6.2-Logical diagram symbols of the buffer amplifier which is logically an invertor without time delay.
This circuit is also assembled on a 18-pin 90- by 110-mm printed card with metal frame. Pin numbers are written in the diagrams.

framework for describing the design of computers utilizing binary computer elements. An algebraic statement of the interrelation of binary variables is a switching function. A function or set of functions is complete if all other functions can be formed from them. The sheffer function ($|$) and Peirce function ($\downarrow$) are of themselves complete. There are indeed such elements which perform these functions. Pairs of functions which are complete are AND ($\cdot$) and NOT ($\bar{}$), OR ($\vee$) and NOT, AND and exclusive OR ($\oplus$), and OR and exclusive OR. The basic logic circuit of the KT-I performs AND and OR operations at its input gates and provides both ON and NOT output. Thus this decision element is logically complete with its storage capacity.

Switching functions are frequently derived from equivalent tabular or diagrammatic representations. The binary value of a switching function for each combination of binary variables can be tabulated in a truth table. Each row of a truth table is made to correspond to a minterm. The general form for the representation of a function F of n variables by minterms is:

$$F = \bigvee_{i=0}^{2^n-1} a_i m_i ,$$

where a_i s are the functional values which can be obtained from a truth table corresponding to a minterm, m_i. A function expressed in this manner is in canonical form. A switching function composed of terms with an OR, each term being composed of factors with an AND is in normal form or disjuntive form.

Veitch Diagrams (Ve 1, Ka 1, Gr 1, Ph 1)

A very useful form of the truth table has been devised by Veitch (which was a special form of Venn diagram), and a modified form was proposed by Karnaugh. These diagrams, which will here be called Veitch diagrams, contains all the information of a tabulated truth table in a more compact

form, and provide a very quick and easy way for finding the simplest expression of a function by geometrical relationships. This Veitch diagram simplification method was extensively used in the case of logical design of the KDC-I.

The Veitch diagram has a unit square for each minterm of the binary variables.

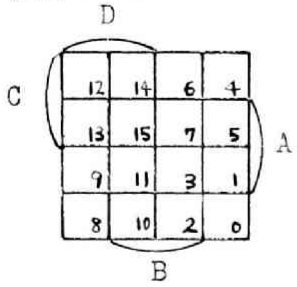
Since the number system of the KDC-I is a binary coded decimal (BCD), the Veitch diagram for functions of four variables was used as a basis. The author prepared diagrams shown in Fig. 6.3, in which diagrams for four, five and six variables are illustrated. In each square is entered the number of the minterm represented by that square, e.g., if the number is 7, it is 0111, then $m_7 = \bar{D}CBA$. Usually m_7 represents $\bar{A}B$ (Ph 1); however, the author chose the above convention since it is possible to obtain a better correspondency between A, B, C,... and weight 1,2,4,... of binary numbers, and a clearer arrangement of numbers in the diagrams. A set of diagrams (1b - 3b) in the figure is used when redundancy of the BCD (which corresponds to 10-15) should be considered.

Simplification of Boolean Functions

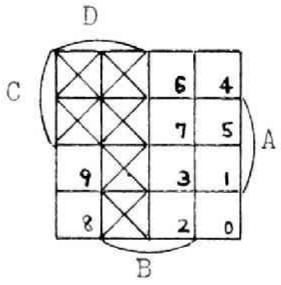
A Boolean function may often be simplified; that is to say, some other expression may be found which represents the same function but may be constructed with less equipment. The input gate of the D flip-flop of the KDC-I permits the operation of functions of second order AND-OR expression or of normal form. It is then possible to count the number of diodes which perform the above operation. Thus as a measure of simplicity the number of diodes necessary is used under the restriction that all simplifications would be in the form of second-order expressions or in the normal form.

The simplification is made by inspection, by Quine simplification

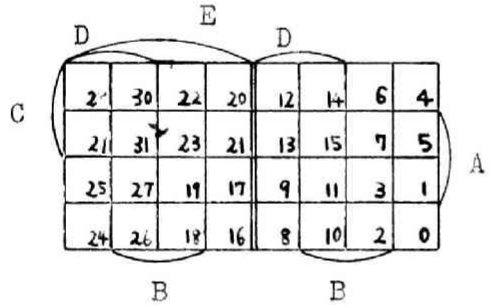
(1a) four variables



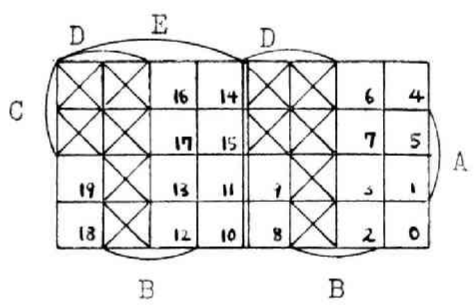
(1b) four variables



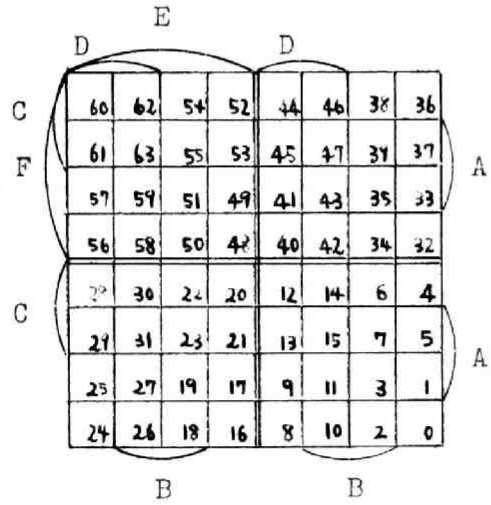
(2a) five variables



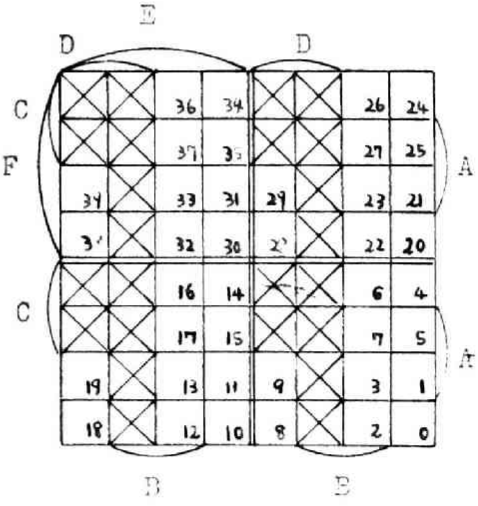
(1b) five variables



(3a) six variables



(3b) six variables



(1a - 3a) : A, B, C, ... ⟶ weight 1, 2, 4, ... of binary number.
(1b - 3b) : A, B, C, D ⟶ weight 1, 2, 4, 8 of BCD number.
E, F ⟶ weight 1, 2 of the second digit, thus 10, 20.
X : redundancy

Fig.6.3-Veitch diagrams in 4,5 and 6 variables. Each square corresponds to a minterm.

method (Q 1, 2), by Harvard simplification method (Ha 1), or by Veitch diagram simplification method (Ve 1). The last method was intensively utilized in the course of the logical design of the computer.

6.4 Veitch Diagram Simplification Method

This method presents a convenient means for the simplification of functions because many of the rules of simplification can be deduced by geometrical consideration. Three important advantages are (Ph 1) : the function to be simplified may be attacked directly, without being expanded into minterm form ; the process of finding prime implicants is simplified by making them correspond to patterns familier to the eye; and the ease with which essential terms may be identified makes it possible to reduce the labor involved in seeking prime implicants.

The systematic approach to obtain the simplest form results in a two-step procedure (Gr 1):

Step 1. Find the essential terms.

(a) Any minterm which is geometrically unrelated to any other minterm is an essential term.

(b) Any minterm which is geometrically related to only one other minterm or group of minterms has the largest such groups as an essential term.

Step 2. Find the best representation for the minterms which are not included in the essential terms. Finding the best representation of minterms not included in the essential terms is sometimes obscured because often there may be more than one equally simple function.
(Each square at the edge should be regarded adjoint to the square at the opposite edge.)

Redundancies

For Veitch diagram simplifications, all minterms which are redundant will have an X placed in their minterm square; all other squares will have functional values of 1 or 0. Redundancies are utilized as 1's whenever their use will simplify a term, or they are ignored otherwise.

In the case of the logical design of the KDC-I, one more step was added at the first stage of the design because of the electrical characteristics of the input gate. The performance margin (chapter 7, 8) of the basic logic circuit of the original type (the diode D_7 in Fig. 7.1 was not inserted at the testing stage of the circuit) deviated considerably due to the operating states of a OR gate. The performance margin when only one of AND gates sends binary 1 to the OR gate differs from that when more than one AND gates send binary 1 signals to the OR gate. The author devised a procedure which solved this problem logically.

> "It is possible to construct any AND-OR input gate in which no more than a single AND gate can send binary 1 at the same time".

Proof : The input gate materializing a function in the minterm cannonical form is such a gate. Any function of n variables can be expressed in the minterm cannonical form.

Next it may clearly be possible to simplify the above gate, e.g., the function $ABC \vee AB\bar{C}$ which is a cannonical form can be simplified as AB. However, the function $A\bar{B}C \vee ABC \vee \bar{A}BC$ should not be simplified as $AC \vee BC$, since the minterm ABC is included in both AC and BC, or if A, B and C equal 1, both AC and BC are 1 at the same time.

Thus the procedure of obtaining the simplest normal form with the above restriction can be stated as follows:

Restriction to Step 1 and Step 2 (it will be called Gate Restriction)

(a) When finding suitable terms in accordance with Step 1 and Step 2, the restriction imposed should be that the same minterm should not be taken more than once.

(b) However, any redundant minterm can be utilized many times, (since signals corresponding to redundant minterms never appear).

The next problem was to what degree the complexity of logic circuits of the computer was increased by this restriction. The author performed the logical design in two ways, i.e., with and without this restriction.

The author was surprised that there was not so much difference of complexity in logic circuits. Most control logic circuits presented no difference, since the job of controlling gate is hardly overlapped among each definite function. Moreover in arithmetic logic circuits it was found that only a few diodes were required extra in materializing the above restriction. This will be shown in section 6.7.

However, in the course of logical design this gate restriction became unnecessary due to the advance of the basic logic circuit. (The diode D_7 in Fig. 7.1 was inserted, and it has the effect of preventing the deviation of margin even if no more than a single gate operate simultaneously.) Since it was at the time when the logical design was almost compoleted, a part of the wiring design was being prepared and there was not much difference of complexity of logic circuits, most parts of the KDC-I logic circuits were constructed under this gate restriction, and some portions were without this restriction.

6.5 Simplification Including Memory Elements

Systematic procedures of finding the minimum number of memory elements for a given logical network were not applied in the case of the logical design of the computer, since practically they are only applicable to

tiny well-defined portions of the computer and the effect o. the simplication is not so appreciable in most forseeable examples, and since the structure of the computer is serial decimal in which most problems could be reduced to the problem of it gate structure and not the problem of the number of memory element (e.g., a one-word register of the computer must have at least 48 memory elements, and by adding proper input gates to them various functions could be performed without the memory element simplification problem).

6.6 Preliminary Design Considerations

The algebraic methods so far described are practically only applicable to tiny well-defined fortions of a computer. Most theories concerning computing machineries treat least requirements for them, e.g., theories concerning the Turing machine are typical examples. Computers actually built are much more complex and useful ones in which convenience is an important factor of the design. Thus the theory of logical design for a whole computer has hardly been developed. The system design of the computer was performed as described in chapter 2. All instructions was defined as described in chapter 3 and in Appendix B. The magnetic tape system was also described in chapter 4. Since the required capabilities of the computer have been specified, the logical design can begin. A computer is usually made up of a number of subunits, the relations of which are investigated early in the design considerations.

The process of the logical design of the KDC-I was divided into several stages.

(1) In the first stage the important scheme of the logical design was decided mainly from the definition of the system, e.g., the type of registers, information paths, the place of an adder and a complementer, the main timing, and so on.

(2) At the same time the logical design of basic arithmetic elements such as a decimal adder, a complementer, and so on was made.

(3) Next the logic of each instruction and the relation among each of them were analysed. The flow chart was made to show a schematic diagram of the logic of instructions.

(4) The logical design was then made piese by piese from the basic instructions.

(5) Time charts were drawn in the course of the design to show the exact time relation of each circuit.

(6) The result of the logical design had been re-examined as much as possible before the adjustment of the computer began.

Before performing detailed designs, a further framework was decided as a result of investigations in which several alternatives were compared and abandoned. The main framework which is at the same time the main features of the logical design will now be presented.

(1) In the floating-point operations, no special exponent register is prepared. The MQ is used for the calculation of an exponent of a floating-point number in the operation.

(2) In all arithmetic operations, the serial addition is used, and all results are placed in the double length 23-digit AC.

(3) In case of an overflow in either fixed-point or floating-point arithmetic operations, no erroneous results will be originated.

(4) When division operation is impossible, the dividend is unchanged, but the control of the sequence is changed.

(5) The operations of the instructions CMP and TLU are made chiefly at the MQ register which is hidden from programmers. These two instructions can be used for both fixed-point and floating-point numbers.

(6) In case of the operations of the instructions PSX, TLU and SCT, the address part of the Order Register OR is used effectively.

(7) From the consideration of timing the logic circuit of the AC has 24 digits. However, the instruction LCS is performed in the 23-digit AC by making a suitable compensation.

(8) In division the non-restoring method is used to get a quotient.

(9) The shifting at the AC is performed quickly by introducing a high speed shifting circuit. This feature is especially useful in shifting and the floating-point operations.

6.7 Design of Arithmetic Elements

The algebraic methods ao far described finds its best application in the design of arithmetic elements. Before describing the details of the arithmetic elements, a block diagram of the arithmetic unit of the computer is shown in Fig.6.4 by referring Fig.2.3 in which main registers and counters, and main information paths are illustrated. The main arithmetic elements are thus adders, subtractors, a complementer, and so on. Some results of the logical design of these circuits will now be presented.

1. 10's Complementer (Fig.6.5 (1),(2),(3))

The subtraction is performed by the addition of an complemented number. In the case where a complementer should be incorporated in a input gate of a register or in a register, the complementer is required to pass a number as it is, to take a 10's complement, or to take a 9's complement. To distinguish these three conditions two memory elements (denoted as E and F) are required. At first both E and F are triggered, and 10's complement is taken until first non-zero digit is detected by the F which as a result is reset, and 9's complement is then taken. (The least significant digit comes first to the input, the second digit next, and so on.) When the most significant digit appears, the operation should be finished, i.e., both E and F (regardless of its state) should be reset.

In Fig. 6.5 (1) is shown the truth table, in which (E,F) = (0,1) is redundant. In Fig.6.5 (2) Veitch diagrams are drawn for obtaining the simplest normal form with the gate restriction. The number of terms is not increased, and only three extra diodes are added to the one (61 diodes) in the simplest normal form without the gate restriction. The logic circuit diagram of this comlementer is shown in Fig.6.5 (3).

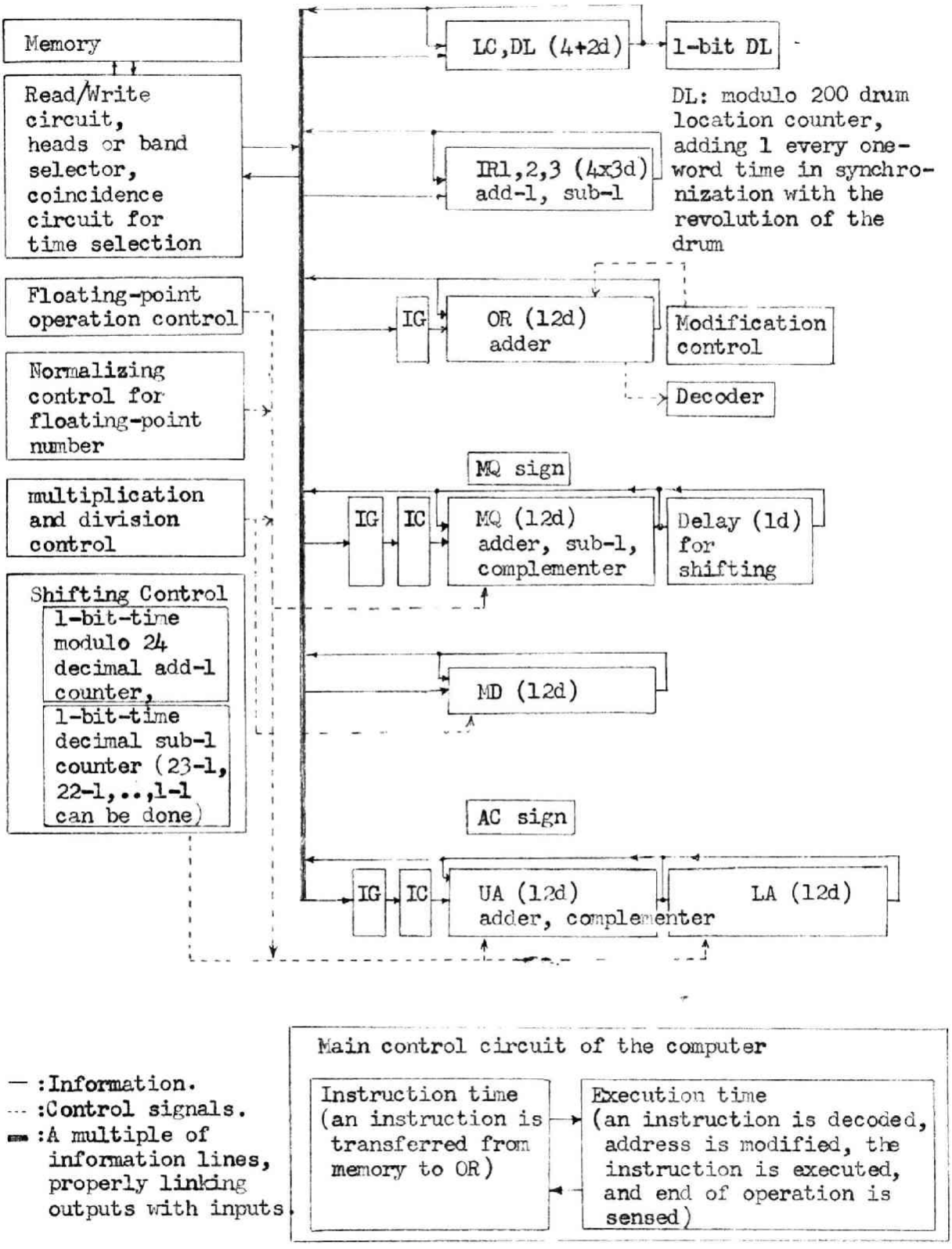
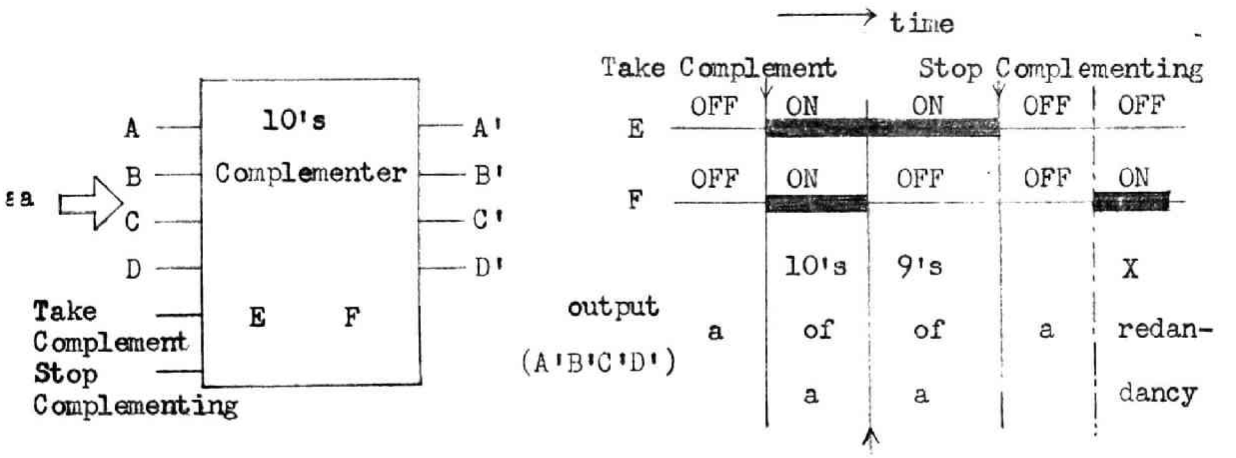


Fig.6.4-Block diagram of the arithmetic unit of the KDC-I.
(12d),(4x3d),etc.: (12 digits), (4x3 digits),etc.
IG : input gate. IC : input complementer.
adder : BCD decimal adder. add-1 : a decimal number can be increased by 1 in one-bit-time. sub-1 : a decimal number can be decreased by 1 in one-bit-time.
A portion of the control unit is also shown.

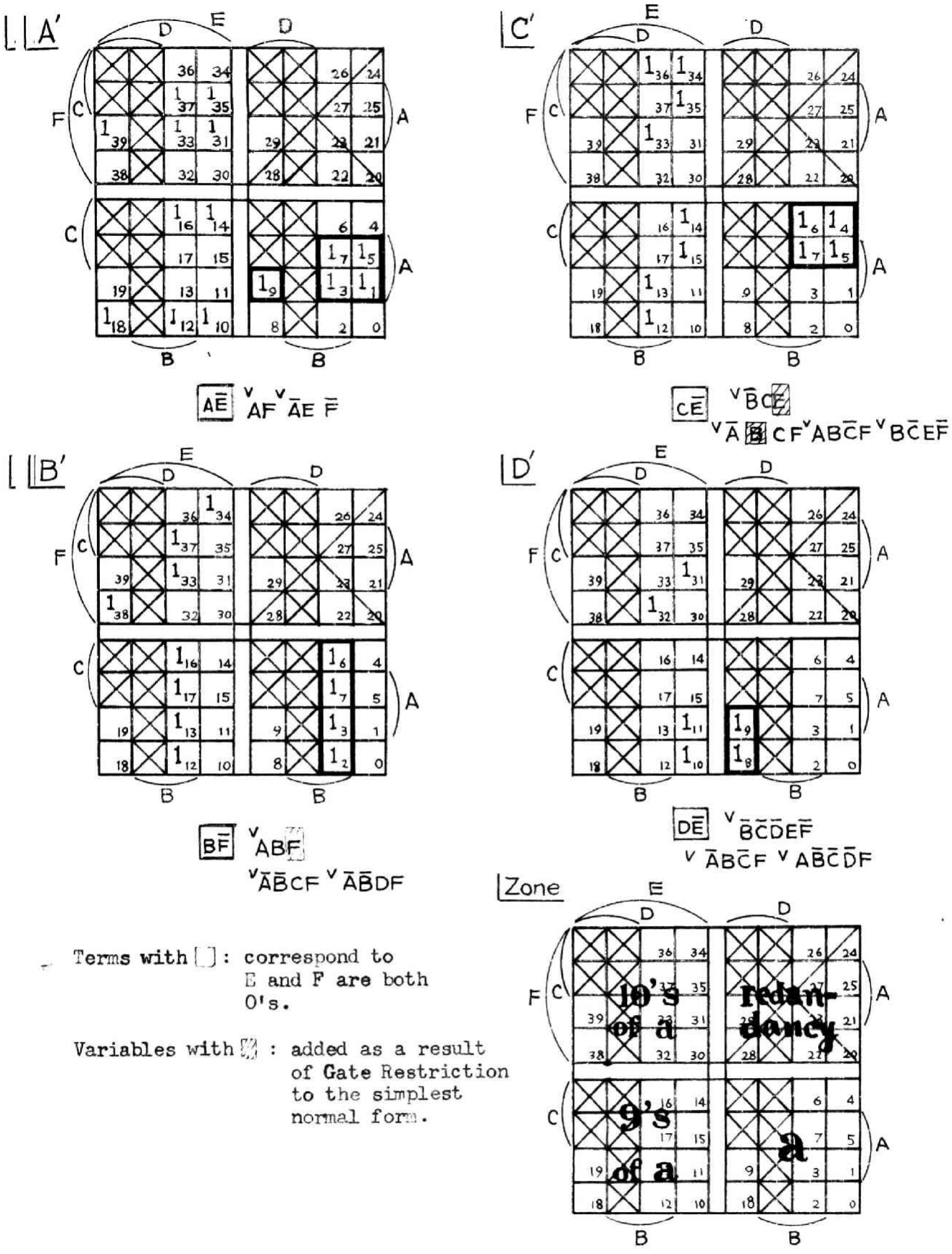


E, F : Memory elements

Input* actual decimal a	EF 00	EF 01	EF 10	EF 11
	a	redan-dancy	9's of a D'C'B'A'	10's of a D'C'B'A'
0	0	X	1 0 0 1	0
1	1	X	1 0 0 0	1 0 0 1
2	2	X	1 1 1	1 0 0 0
3	3	X	1 1 0	1 1 1
4	4	X	1 0 1	1 1 0
5	5	X	1 0 0	1 0 1
6	6	X	1 1	1 0 0
7	7	X	1 0	1 1
8	8	X	1	1 0
9	9	X	0	1

* The least significant digit comes first, the second digit next, and so on.

Fig.6.5 (1) - 10's Complementer; truth table.



Terms with [ ] : correspond to E and F are both 0's.

Variables with ▨ : added as a result of Gate Restriction to the simplest normal form.

Fig.6.5 (2) - 10's Complementer; Veitch diagrams to obtain both the simplest normal forms and the simplest normal forms with the gate restriction.

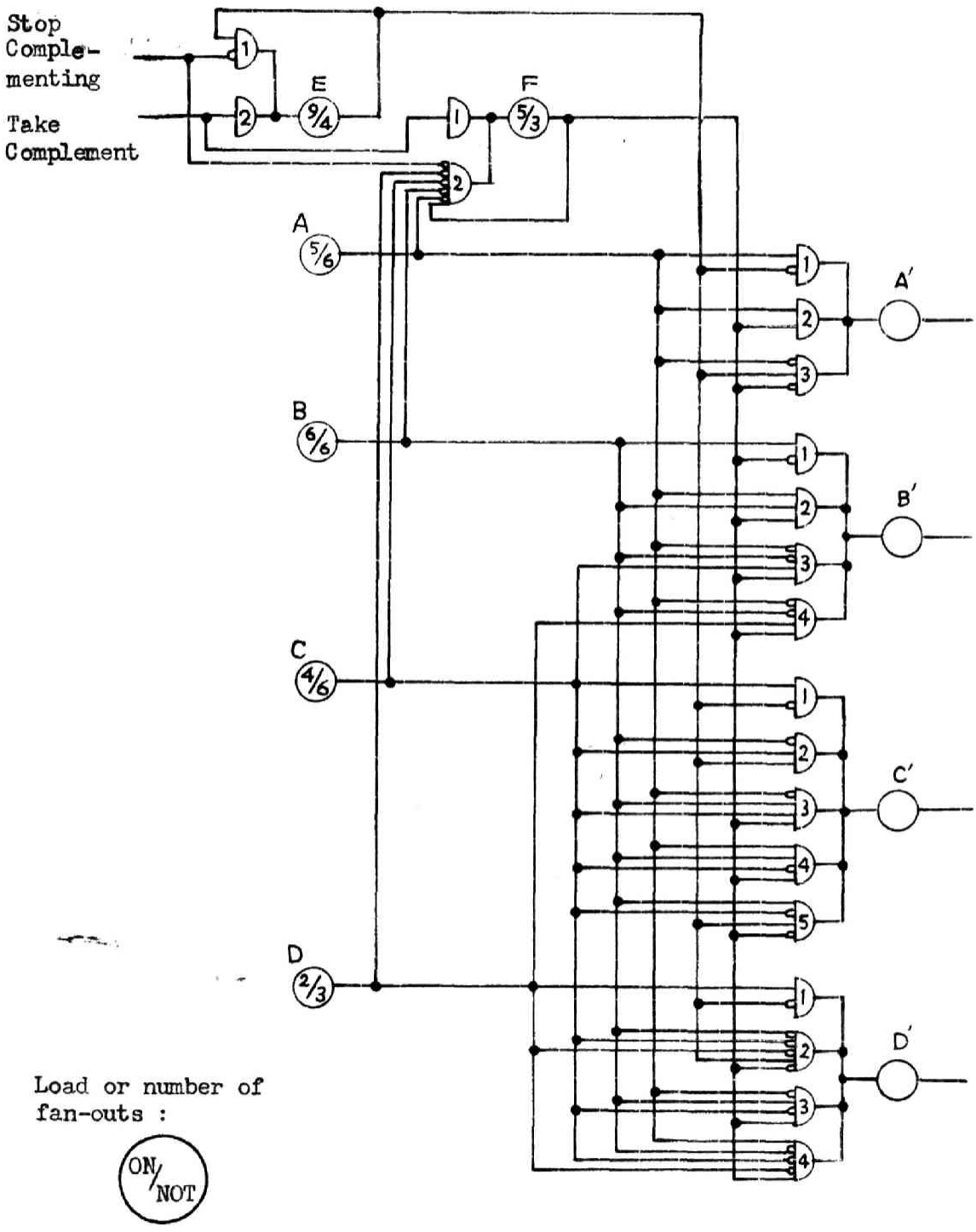


Fig.6.5 (3) - 10's Complementer; logic circuit materializing the function in the simplest normal form with the gate restriction.
(actually used in the KDC-I)

2. 1-bit-time Modulo 24 Decimal Add-1 Counter (Fig.6.6 (1),(2.1),(2.2),(2.3),(3))

This counter is used for control of the right shift of the 23+1 digit AC, since n-digit right shift can be performed by 24-n digit left shift. Again the gate restriction causes only a slight increase of the number of diodes, and no increase of the number of terms.

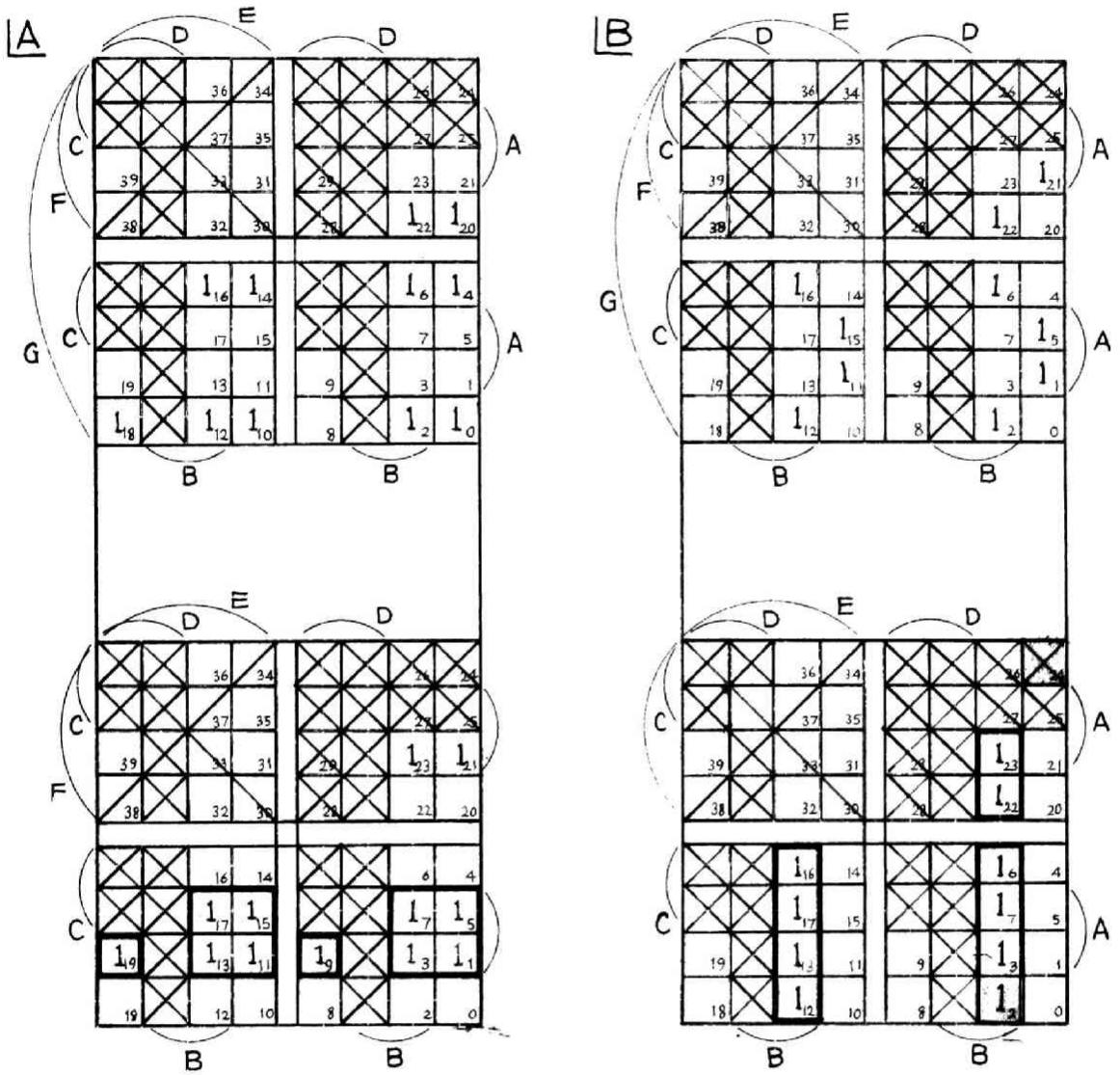
3. Range Detector ($0 \leq |\varepsilon| \leq 17$) (Fig.6.7)

There are many logic circuits which are difficult to follow its logic at a glance. The circuit shown in Fig.6.7 is not very tricky, however, it would be an example which belongs to the above category. This circuit is used to detect the range of the difference of characteristics in the case of floating accumulation operations.

When the primary design was completed, the destribution of the number of D flip-flops of the KDC-I when the number of their ON and NOT fan-outs are taken as variables was surveyed. These data together with the final ones are shown in Fig.6.8.

	Add.1 G	Actual Sum.	Weights F 20 E 10 (B_2 A_1)	D 8	C 4	B 2	A 1
0	1	1					1
1		2				1	0
2		3				1	1
3		4			1	0	0
4		5			1	0	1
5		6			1	1	0
6		7			1	1	1
7		8		1	0	0	0
8		9		1	0	0	1
9		10	1				0
10		11	1				1
11		12	1			1	0
12		13	1			1	1
13		14	1		1	0	0
14		15	1		1	0	1
15		16	1		1	1	0
16		17	1		1	1	1
17		18	1	1	0	0	0
18		19	1	1	0	0	1
19		20	1 0				0
20		21	1 0				1
21		22	1 0			1	0
22		23	1 0			1	1
23	1	0					0

Fig.6.6 (1) - Modulo 24 decimal Add-1 counter; truth table. This counter is used for control of the right shift of the 23 + 1 digit AC, since n-digit right shift can be performed by 24-n digit left shift.
Add-1 counter: the contents can be increased by 1 in one-bit-time.



$A^{t_1} = \boxed{A\bar{G}} \vee \bar{A}G$

(= also 2nd order Minimal Form or simplest normal form.)

Load (on/not)

$B^{t_1} = \boxed{B\bar{G}} \vee A\bar{B}\bar{D}G \vee \bar{A}BG$

$(= \boxed{B\bar{G}} \vee A\bar{B}\bar{D}G \vee \boxed{\bar{A}B}$ 2nd Order Minimal Form)

A	B	C	D	E	F	G
6/6	6/3	4/1	5/2	4/1	3/1	14/6
2nd M.F 6/6	4/3	4/1	4/2	4/1	3/1	6/6 G

Fig. 4.4 (2.1) - Module 24 decimal Add-1 counter: Veitch diagrams.

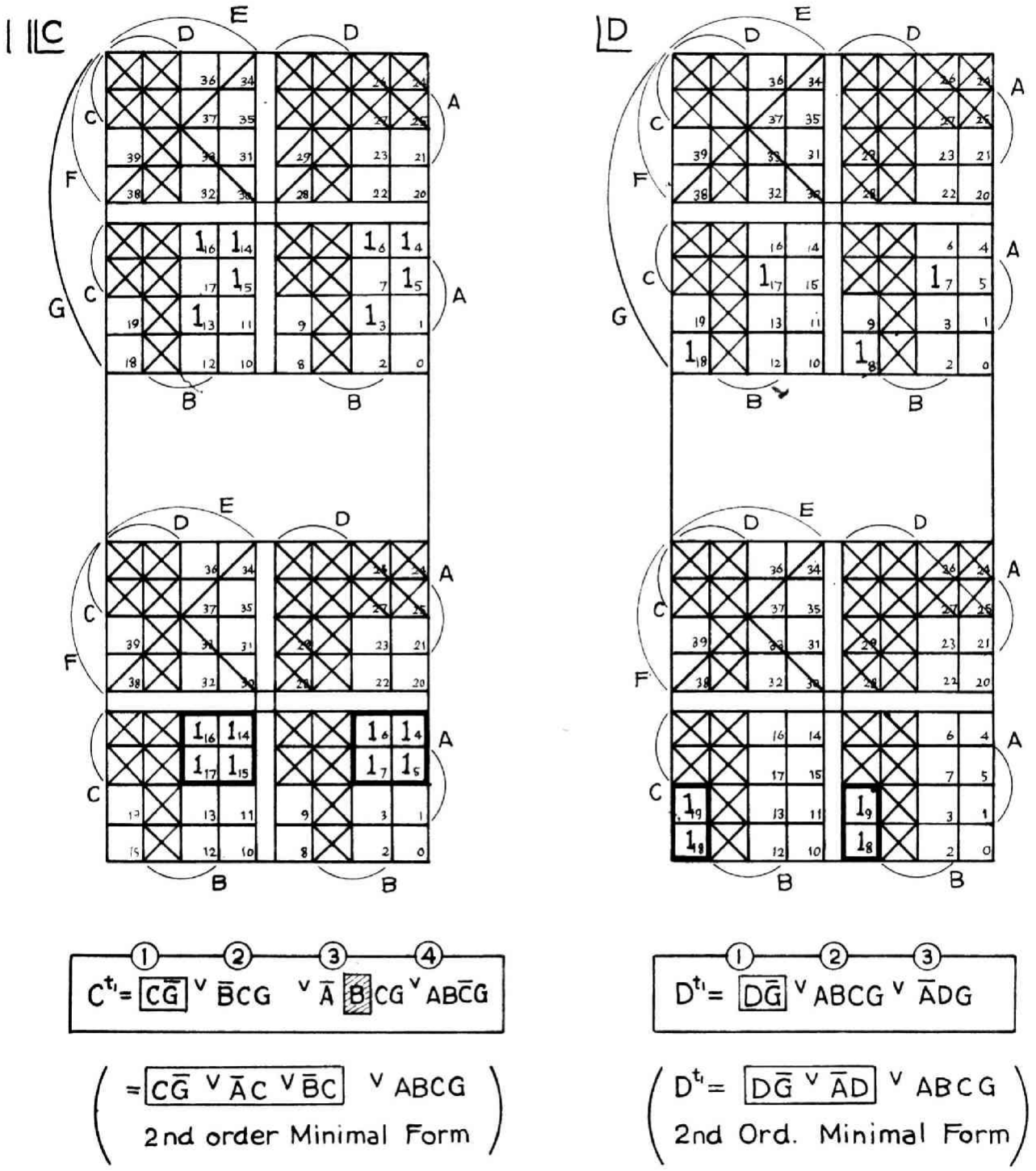


Fig.6.6 (2.2) - Modulo 24 decimal Add-1 counter; Veitch diagrams.

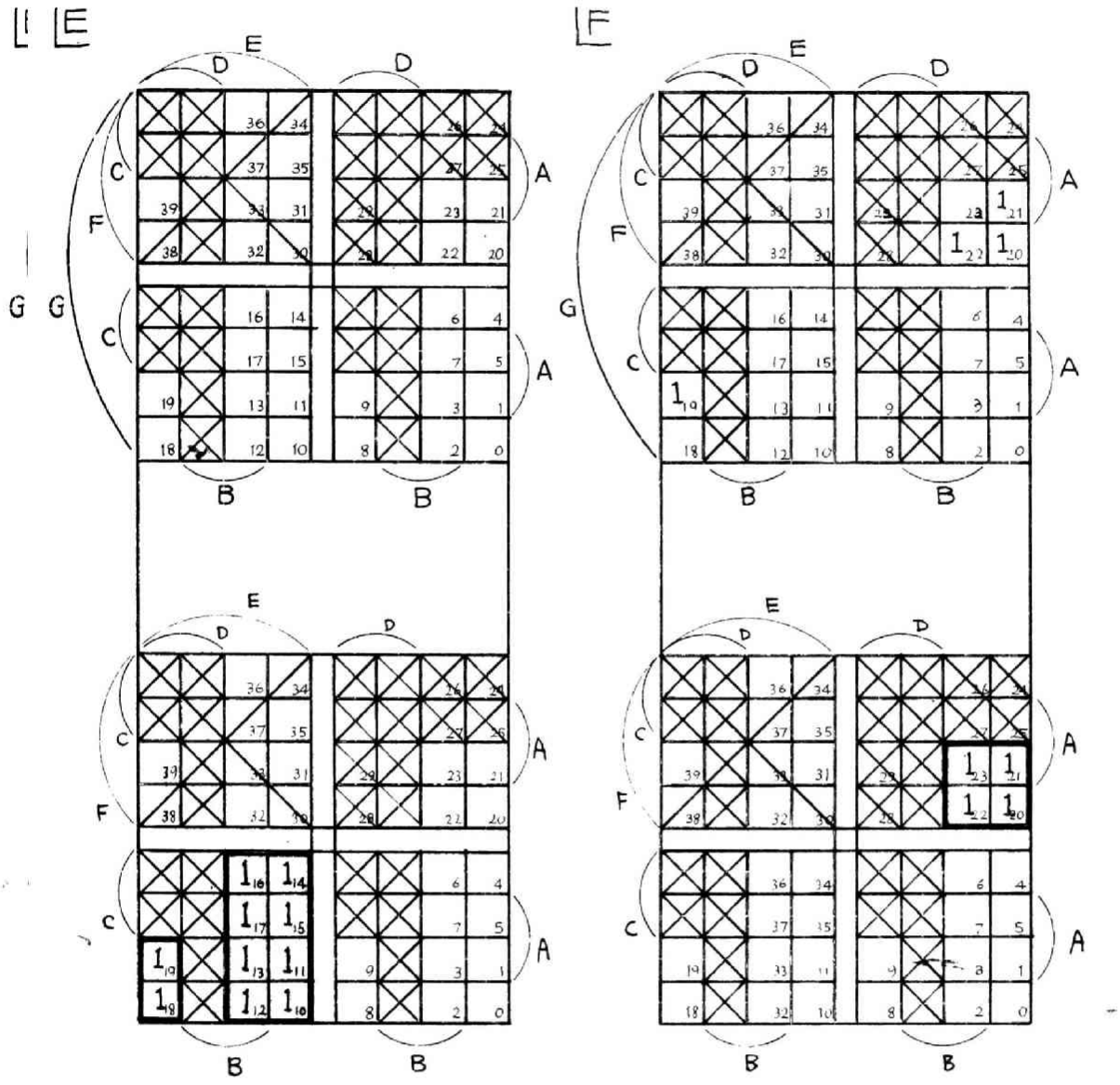


Fig.6.6 (2.3) - Modulo 24 decimal Add-1 counter; Veitch diagrams.

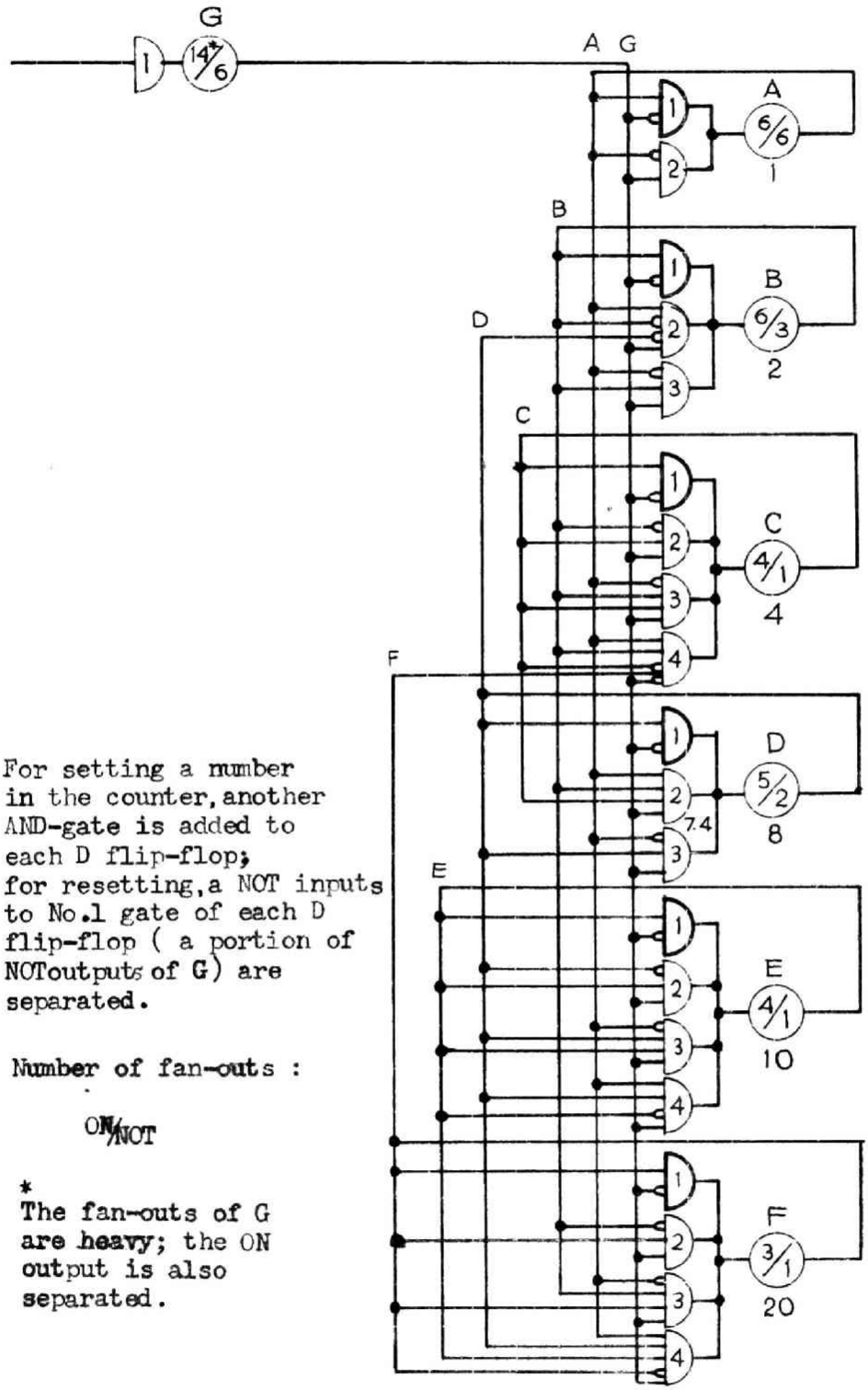
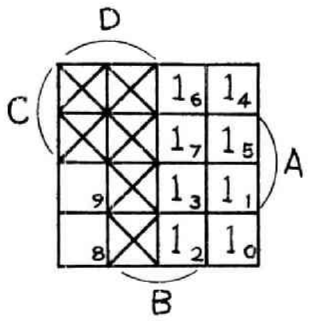


Fig.6.6 (3) - Modulo 24 decimal Add-1 counter; logic circuit incorporating the function in the simplest normal form with the gate restriction. (actually used in the KDC-I)



0,1,...,7 : $\bar{D}$

v m 10
0 0 0
0 0 1
0 0 2
0 0 3
0 0 4
• • •
• • •
• • •
0 0 9
0 1 0
0 1 1
0 1 2
• • •
• • •
• • •
0 1 6
0 1 7

Three-digit information (difference of characteristics) or v, m and 10 appears at MQ flip-flops in the following way;

	T_0	T_1	T_2
MQ1A~D	m	v	-
MQ2A~D	10	m	v

Timing: $T_0\ T_1\ T_2 \cdots T_{11}\ T_0$ ... (every one-bit-time), thus T_i comes every one-word-time.

A,B,C and D correspond to the weight 1,2,4 and 8 of BCD number.

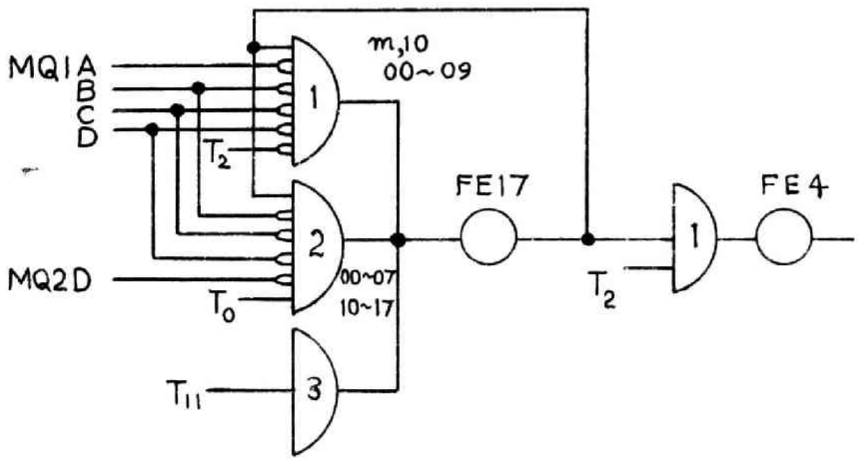
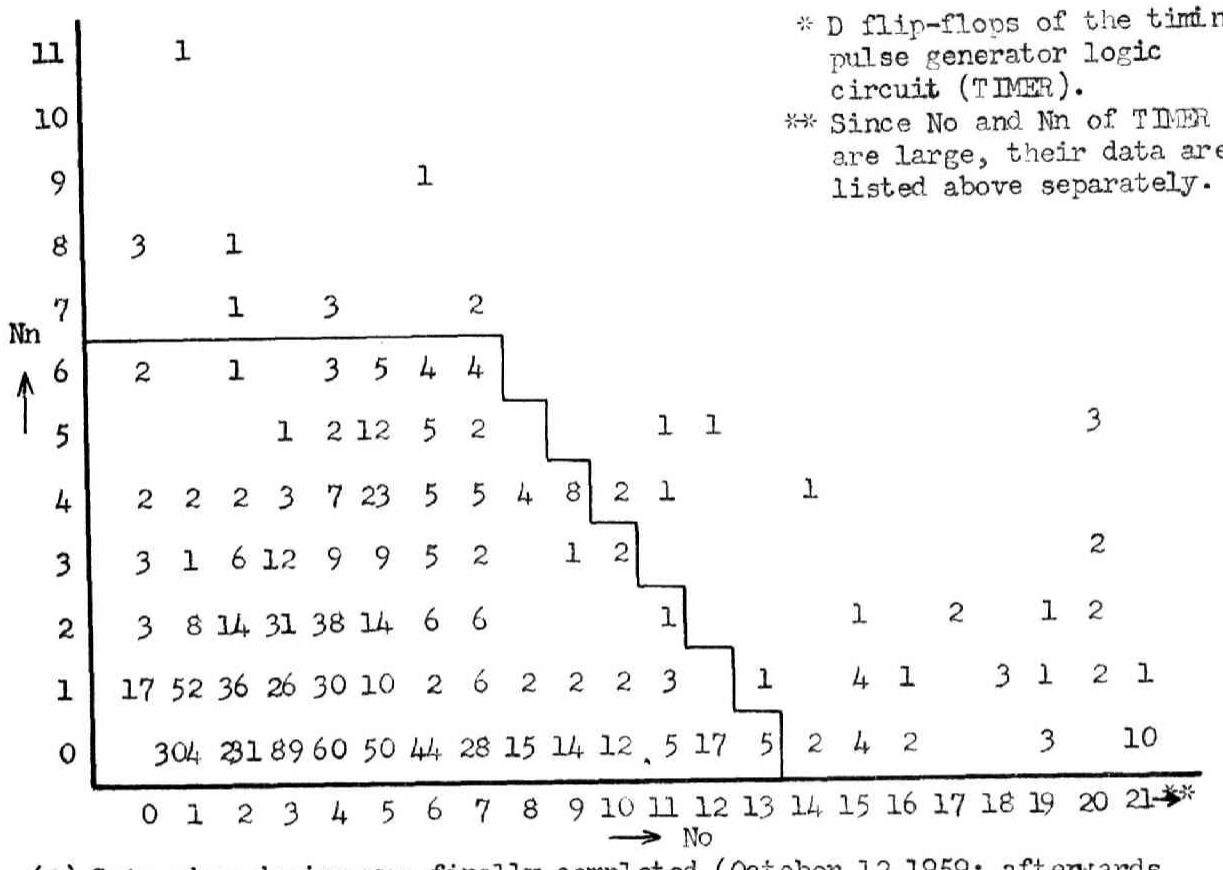


Fig.6.7-Range detector ($0 \leqq |\varepsilon| \leqq 17$). This circuit is used to detect the range of the difference of characteristics in the case of floating accumulation operations.

(1) Data when primary design was completed (July 31,1959).

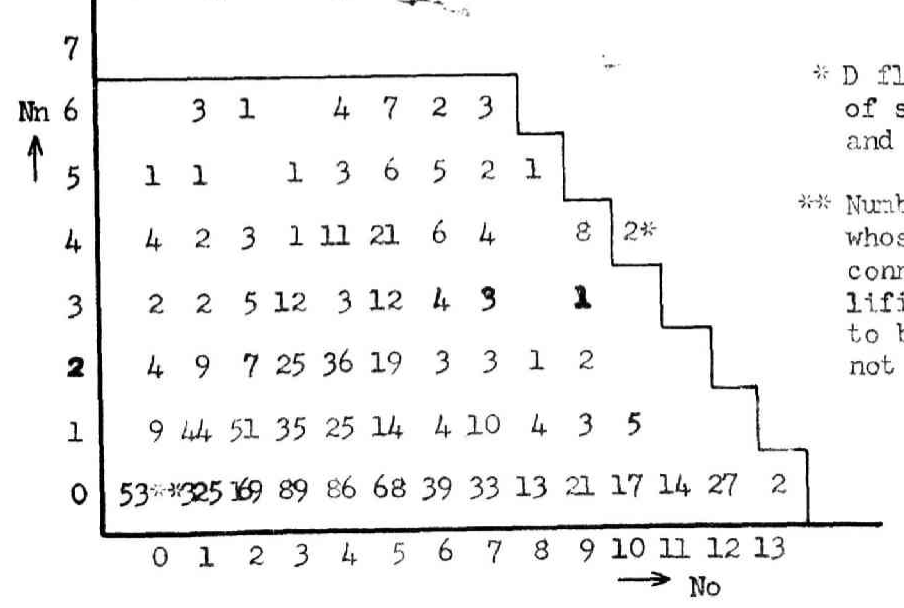
Name	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	*
No	104	67	34	31	33	33	41	32	57	55	67	94	
Nn	35	19	17	12	15	14	21	7	22	19	24	52	



* D flip-flops of the timing pulse generator logic circuit (TIMER).

** Since No and Nn of TIMER are large, their data are listed above separately.

(2) Data when design was finally completed (October 12,1959; afterwards there was a slight change of data due to corrections of logic circuits during the adjustment).



* D flip-flops for borrow of sub-1 counter at MQ and IRs.

** Number of D flip-flops whose outputs are solely connected to buffer amplifiers. Fan-outs connected to buffer amplifiers are not counted in No and Nn.

Fig.6.8 Distribution of the number of D flip-flops of the KDC-I when the numbers of their ON and NOT fan-outs (No and Nn respectively) are taken as variables.

Chapter 7

CIRCUITS OF THE COMPUTER

7.1 Introduction

The operating performance of electronic circuits, especially of basic logic circuits, used for a computer plays an important role in the overall performance of the computer. Characteristics of basic logic circuits provide a limit of the performance of a system which uses them. A drastic improvement of the computation speed could only be achieved by developments of new basic logic circuits. A basic logic circuit of the KDC-I operates at 230 kc/s, while today a new element at the developmental stage is reported to operate at several hundred kc/s, which hints a possibility to construct a very high speed computer.

Another important factor is reliability, since many basic logic circuits are used for a computer and the computer must exhibit very high reliability, e.g., a single failure could introduce errors which invalidate the final solution.

At the time of the design of the KDC-I several computers which used vacuum tubes, parametrons or transistors had already been constructed at a few research laboratories in this country. However, some of them were reported to be having trouble concerning the reliability. There was not enough time to develop a better basic logic circuit at that time, and decision was made to use an element which had proven characteristics and reliability. A transistorized basic logic circuit developed by the Electrotechnical Laboratory of Japan (ETL type logic circuit)(Ni 1), which is a transistor version of the basic logic circuit used for the SEAC computer of National Bureau of Standard (N.B.S.) of U.S.A. which uses the dynamic circuitry techniques (El 1), was chosen as a basic logic circuit of the KDC-I after comparing the operation speed, flexibility in logical design,

and so on with those of parametrons.

The overall operational reliability of a digital computer depends upon design techniques, operating and maintenance techniques. The achievement of reliability was a goal that was persued from the very beginning of the design. Characteristics of the DTL type logic circuit was fully studied, and some modification to the circuit was made to improve its characteristic. Next, efforts were exerted to minimize the deviation of characteristics of each basic logic circuit. The performance margin of all basic logic circuits used for the computer was measured and the output pulse width of each circuit was adjusted. Moreover, the same kind of margin was measured on a group of logic circuits which forms each registes of the computer. After the completion of the computer, marginal checking is periodically done to verity the performance of the computer. A new unit to display the performance margin of the basic logic circuit which was suggested by the author was constructed, the details of which will be discribed in chapter 8.

About 5,000 transistors and 20,000 diodes have been put into use. Various electronic circuits were designed. The main circuits are: the transistorized basic logic circuit, clock pulse supply circuits, buffer amplifier, read and write circuits and band decoder circuits for the magnetic drum, circuits to connect the I/O components, lamp circuits, switch circuits, circuits related to magnetic cores, circuits related to magnetic tapes and power supply circuits.

In this chapter, the basic logic circuit of the computer, the performance margin of the basic logic circuit and of a group of basic logic circuits will be described in the first place. Next, a buffer amplifier and some other circuits which are thorght interesting will be described.

7.2 Basic Logic Circuit of the Computer

The dynamic circuitry techniques was first adopted by the National

Bureau of Standards (N.B.S.) of U.S.A. for its SEAC and DYSEAC computer, though the idea was suggested in the preliminary report on the UNIVAC (El 1). Although this is a vacuum tube version, its principle is very attractive as a basic logic circuit. A transistor version was reported by J.H.Felker in 1952 (Fe 1). Another transistor version was suggested by S. Takahashi et al. of Electrotechnical Laboratory of Japan. After carefully studying the performance of this ETL type logic circuit, some modification was made to improve its characteristic, and this modified circuit was used as a basic logic circuit of the computer. After a time, the Computer Control Co. of U.S.A. has also announced the one which operates at 1 Mc/s (Pr 1).

The logical property of the basic logic circuit of the computer is of the delay memory element or the D flip-flop with input gates which perform logical operations (Ph 1).

In Fig. 7.1 the circuit diagram of the basic logic circuit is presented. Since most of the circuit operations are the same as that of the original ETL type logic circuit (Ni 1), only the principle and a new feature of this circuit are given. The logical AND and logical OR are performed by diodes at the input. There are two kinds of packaged basic logic circuits: type A6 and A7. There are also three kinds of packaged diode (-resistor)-gate circuits : type D8, D9 and D11. The circuit diagrams of the type D packages and their logical diagram symbols are shown in Fig. 7.2. Logical properties of diode gates of the type A and D packages are as follows:

Type A6 package : $X_1 \ldots X_4 \cup X_5 X_6 \cup$ (outputs of the type D packages),

Type A7 package : $X_1 \ldots X_6 \cup$ (outputs of the type D packages),

Type D8 package : $X_1 \ldots X_5 \cup X_6 \ldots X_{10}$, $X_{11} \ldots X_{15}$ (it has two independent outputs),

Type D9 package : $X_1X_2X_3$, $X_4X_5X_6$, $X_7X_8X_9$, $X_{10}X_{11}X_{12}$
(it has four independent outputs),

Type D11 package : $X_1 \ldots X_6$, $X_7 \ldots X_{12}$, (it has two independent outputs),

where Xi(i=1,2,...) is an input logical variable.

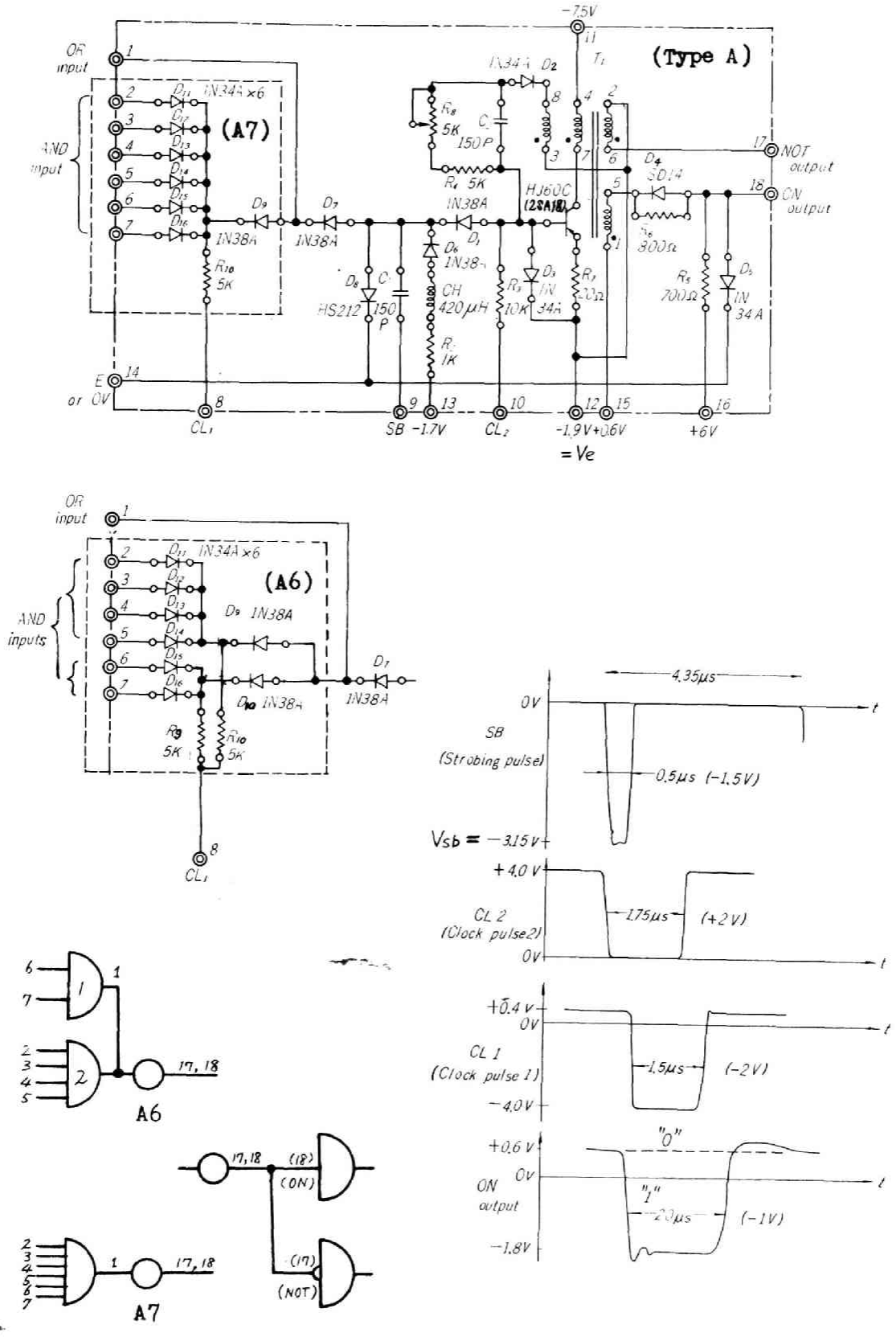


Fig.7.1-Circuit diagrams of the basic logic circuit of the KDC-I and their logical diagram symbols.

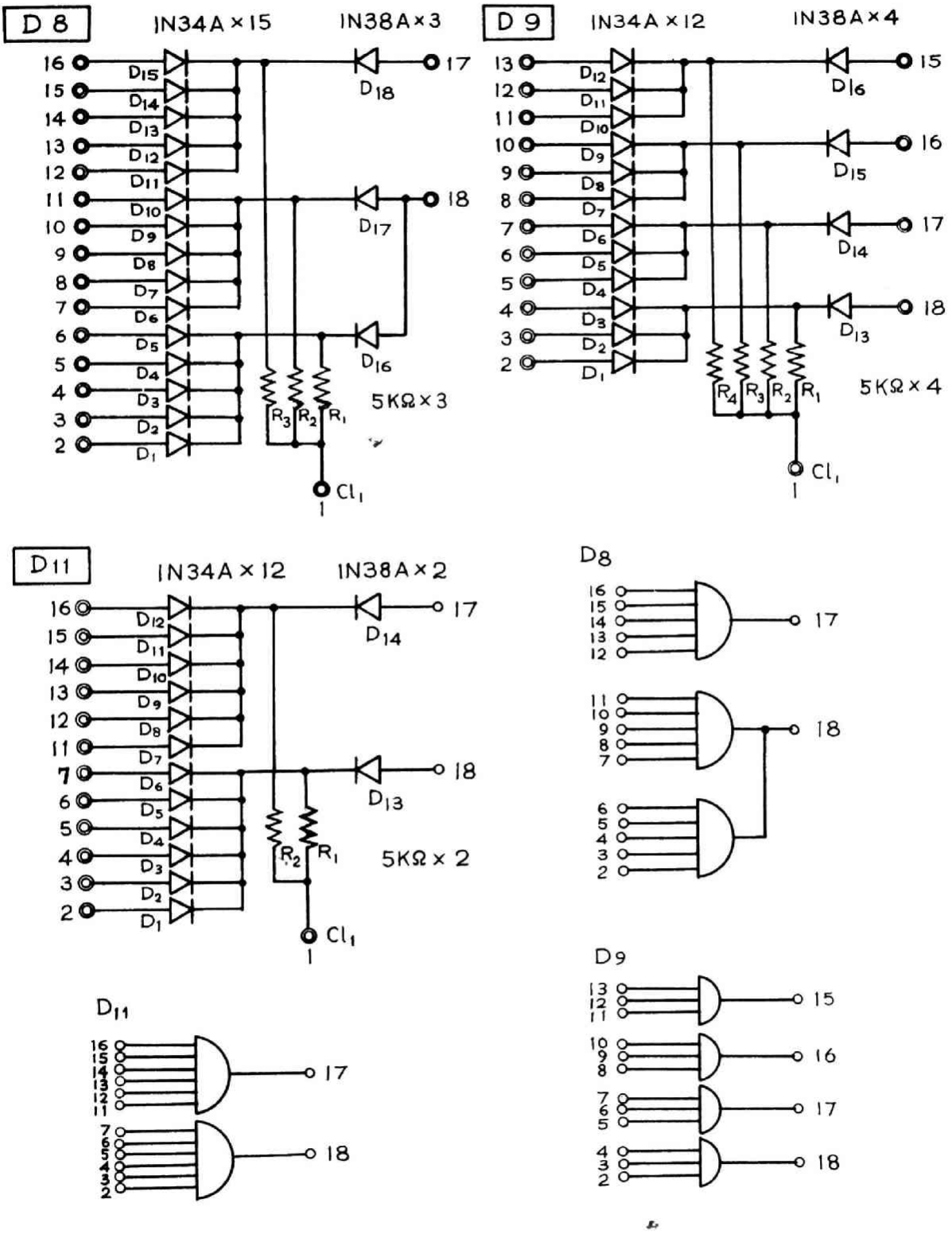


Fig.7.2-Circuit diagrams of the type D packages and their logical diagram symbols.

By connecting pins of the A and D type packages properly, it is possible to construct a logic circuit of various complexity. All logic circuits of the KDC-I are constructed from these five kinds of packages. The maximum available number of fan-ins and fan-outs of the basic logic circuits was decided first from circuit analyses (Ni 1) and finally by referring the results of experiments. However, the logical design had actually been conducted under rather severe restrictions on the maximum available number of fan-ins and fan-outs of the basic logic circuit so as to get a wider performance margin of the circuit. The actual condition for the fan-ins or this AND-OR (second-order expression) gate circuit is as follows:

AND - gate : no more than 6 inputs (to one OR-gate input),

OR - gate : no more than 8 inputs (to the transistor amplifier).

The actual condition of fan-outs of the A type package (which has both ON and logical NOT outputs) is :

$No + Nn \leqq 13$,

$6 \geqq Nn \geqq 0$ and $No \geqq 0$,

Where No is a number of ON outputs and Nn is a number of NOT outputs.

Each type of circuits is assembled on a plug-in type printed card and is packaged with a metal frame. In Fig. 7.3 views of the type A7 package are shown.

The output from the diode-gate circuit or one bit of information is temporarily stored in the capacitor C1 (or delayed). If one bit of information is a binary "1", a negative puke is applied to the C, and the C1 is charged. If it is a binary "0", a pulse is not applied and the C1 is not charged. Next, by strobing pulse, the charge of the C1 is tested and reset. If the C1 is charged or a binary "1" is stored, the strobing pulse triggers the transistor one-shot bloking oscillator; if the C1 is not charged, it is not triggered. The output impedance should be kept low since it is

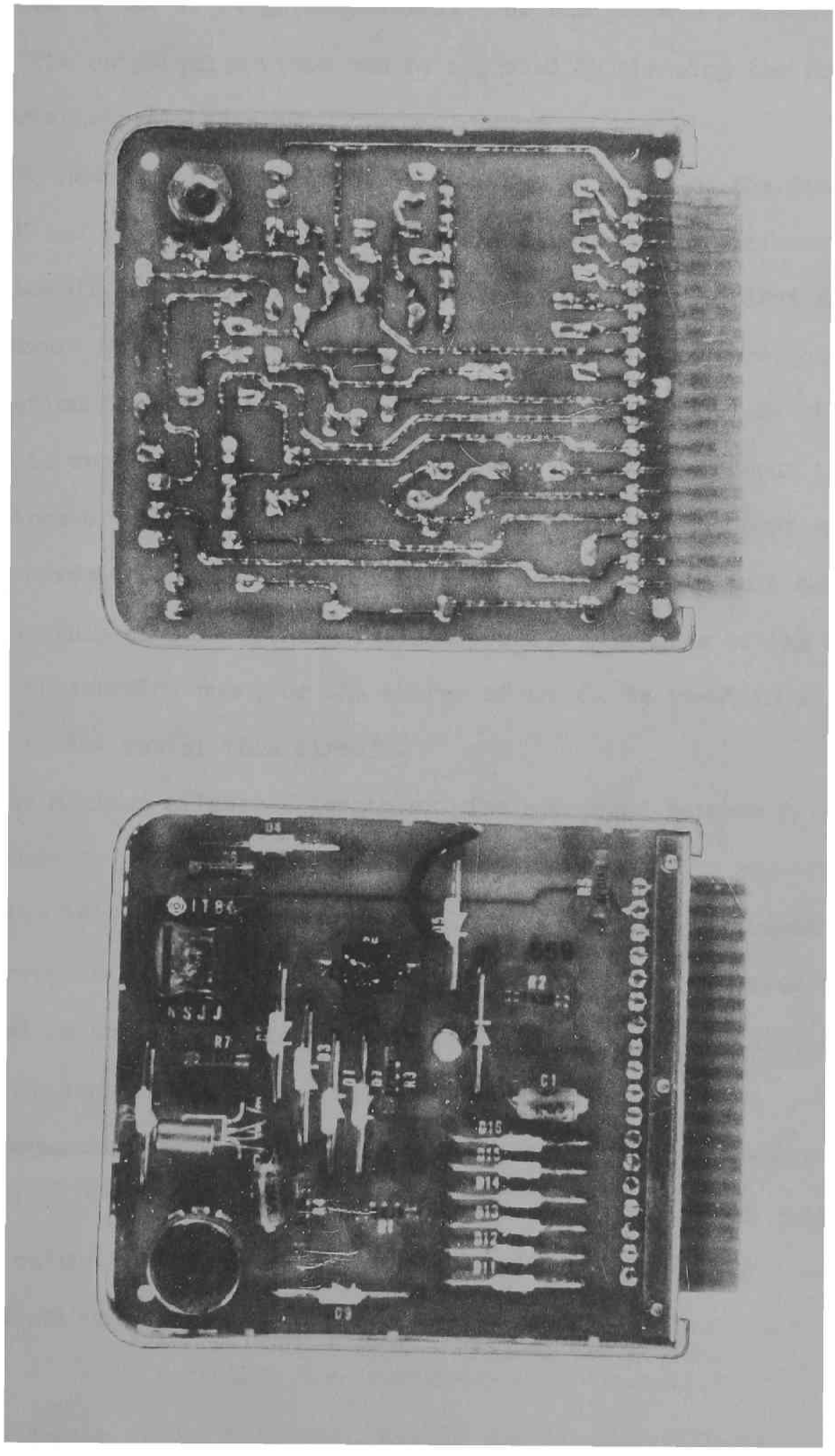


Fig.7.3-View of type A7 package (basic logic circuit of the KDC-I). Eighteen-pin 90- by 110-mm card, with metal frame.

connected to the diode-gate circuits. For this purpose a transformer is used. The output pulse width can be adjusted by choosing the proper value of the variable resistance R8.

The new feature of this circuit is the addition of the diode D6, the coil CH and the resistor R2 across the terminals of the information storage capacitor C1 and -1.7V, which was developed by Prof. T. Sakai and Prof. H. Hagiwara of Kyoto University. As described in the above, one bit of information is temporarily stored in the C1, then tested, and the charge of the C1 is reset. This process is repeated. However, without this circuit, the charge of the C1 is not satisfactorily reset and the performance margin of the circuit depends on a pattern of input pulse train to a considerable degree. This circuit greatly facilitates the discharge of the C1 at the end of the strobing pulse, or the charge of the C1 is reset to a satisfactory degree by the use of this circuit.

The maximum allowable length of wire connected between input and output terminals of basic logic circuits were determined by experiments. The connectability of terminals of a female connector for a circuit package or the appropriate maximum number of wires which can be connected to the terminal is two as a rule. Thus in the case of the KDC-I all terminals or nodes are connected with one line.

The maximum allowable length for the ON, NOT or diode-gate circuit output (which is connected to the OR input) was decided as follows :

for ON output : actually no restriction within the racks of the computer,

for NOT output : approx. 200cm in the case of the ordinary wire,
approx. 500cm in the case of the low-capacitance wire
(actually red insulation tube is covered),

for diode-gate output : approx. 30cm in any case.

7.3 Performance Margin of the Basic Logic Circuit

The marginal checking was developed by Lincoln Laboratory of M.I.T.(Gr 1)

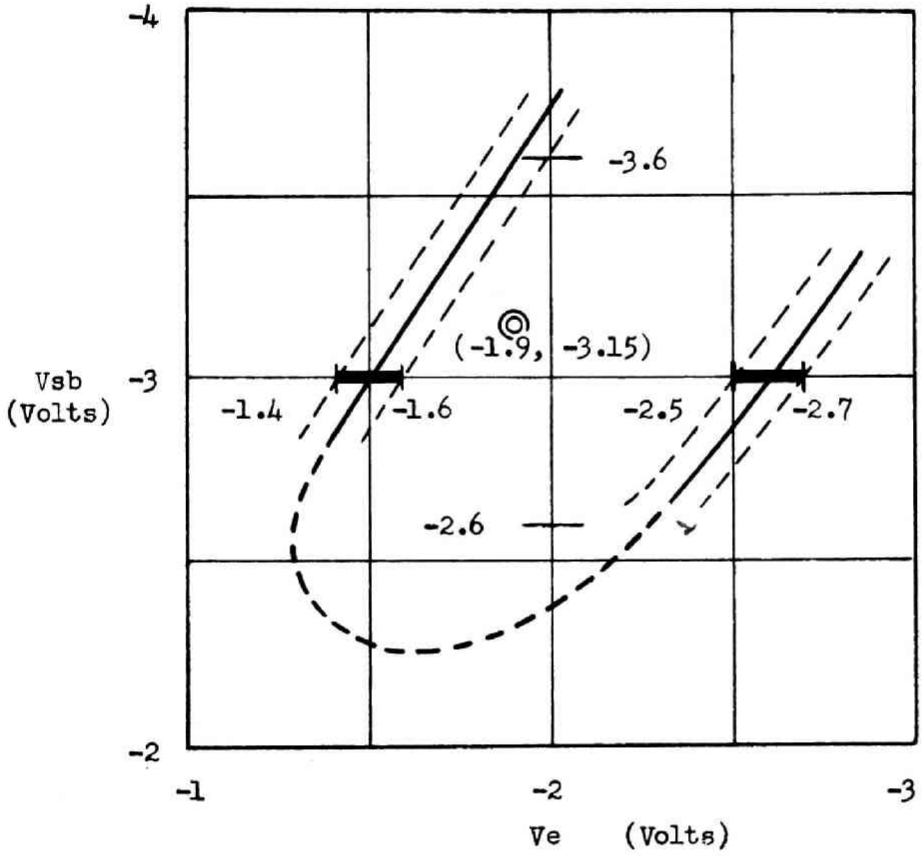
and it has been extensively used in the disign phases of computing systems, as well as in day-by-day operation of such systems.

In the case of the basic logic circuit the supply voltage for the transistor emitter (Ve) and the peak voltage of the strobing pulse (Vsb) are considered to be the two most important marginal checking parameters of the circuit (though in reality there are many parameters that vary simultaneously), since they are related to the most delicate functions of the circuit. Thus, in practice, the following measurement is performed. A train of pulses is applied to the input of a basic logic circuit, and values of these voltages as marginal checking parameters are varied, then the region of satisfactory operation and the region of unsatisfactory operation are measured by observing the waveform of the output pulse trains on the screen of an oscilloscope. Results obtained are plotted on graphs in which the Ve is taken as abscissa and the Vsb as ordinate. This Ve-Vsb graph can also be displayed directly on the screen of a cathode-ray tube by the display unit which is described in chapter 8.

The Ve-Vsb graph is important, since the operating states of most components of basic logic circuit, e.g., resistors, condensers, diodes and the transistor, can be learned from it. Thus the Ve-Vsb graph will be called the performance margin of this basic logic circuit. The graph is used as the basis for the selection of basic logic circuits.

In Fig. 7.4 a typical Ve-Vsb characteristic or performance margin of the basic logic circuit and the criteria for the selection of basic logic circuits are presented. All basic logic circuits are selected under these criteria. It is hoped that any logic circuit which consists of many these basic logic circuits will have the margin whose boundary is the inner envelop of the boundaries of margins of these basic logic circuits. In reality, however, it becomes smaller than that expected, since there would be much noise on dc power supply lines, much deterioration in waveforms,

(Typical performance margin)



(Criteria for the selection of basic logic circuits)

1. The output pulse width at -1V level should be adjusted to 2.05 us when Ve = -2V and Vsb = -3V.

2. When Vsb = -3V, Ve margin must be (-2.6 ± 0.1 V — -1.5 ± 0.1V).

3. When Ve = -2V, Vsb margin must be no less than the range (-3.6V — -2.6V).

Fig.7.4-Typical performance margin of basic logic circuits and their selection criteria.

and so on. The performance margin of the MD of the KDC-I which consists of four 12-bit ring counters, in the case when only one pulse recirculates in each counter, is shown in Fig. 7.5. The result is satisfactory in the sense that the overall margin is not deteriorated from that of a single circuit. The same kind of margin was measured on all registers and counters of the KDC-I. The results were not so different from that shown in Fig. 7.5, except that of the MQ. Fig. 7.6 shows the performance margin of the MQ, four 12-bit ring counters with a decimal adder, a complementer and an add/sub.-1 counter, consisting of about 60 basic logic circuits. It indicates that in the case of the addition at the MQ the margin was greatly decreased, although there is a result that the addition at UA does not appreciablly affect the overall margin of the UA. It was, however, not possible to trace the cause at that time.

After the completion of the computer the performance margin was measured while one hundred places of natural logarithm e were being computed repeatedly. The results were :

Ve margin: -1.7V ~ -2.2 V, when Vsb = -3,15V

Vsb margin: -3,0V ~ -3,25V, when Ve = -1.9V

Since it is not simple to vary the value of the Vsb and since it is possible to estimate the approximate Vsb margin when Ve margin is known, only Ve margin is measured once in a month as the marginal checking of the computer. The Vsb is kept at -3.15V and the Ve -1.9V in day-by-day operation of the computer. The logic circuit of the computer consists of about 1400 Type A packages, about 400 Type D packages and about 120 buffer amplifiers (which are described in the next section) as described in chapter 6. The torelance of the Ve of a single basic logic circuit when Vsb = -3.15V is approximately 1.1V, while it is 0.5V in the case of a logic circuit consisting of about 1400 basic logic circuits.

The Ve margin varies with time, two-yeardata of Ve margin are shown

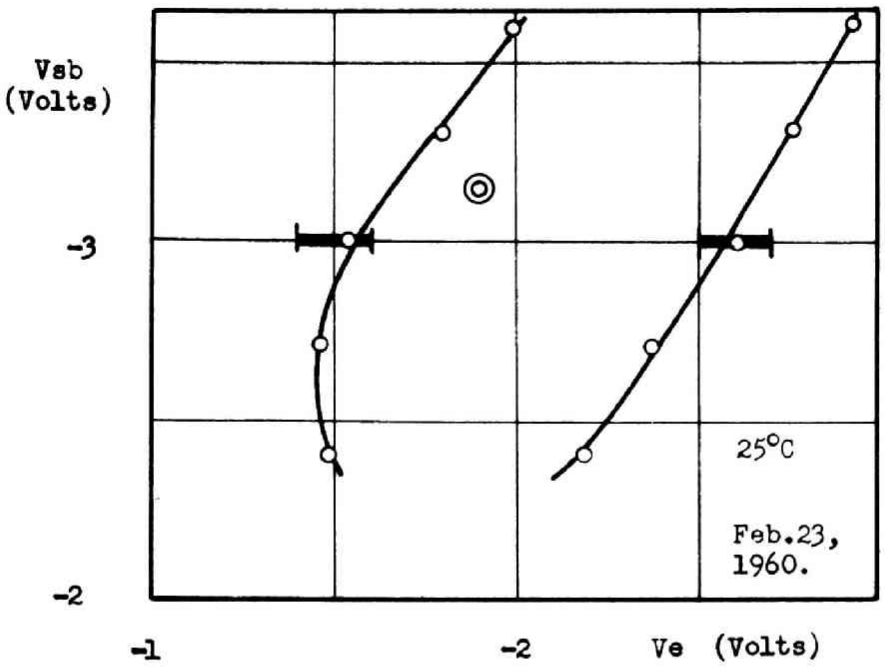


Fig.7.5-Performance margin of the MD of the KDC-I (four 12-bit ring counters, consisting of 48 basic logic circuits). Measurement was performed when only one pulse recirculated in each counter.

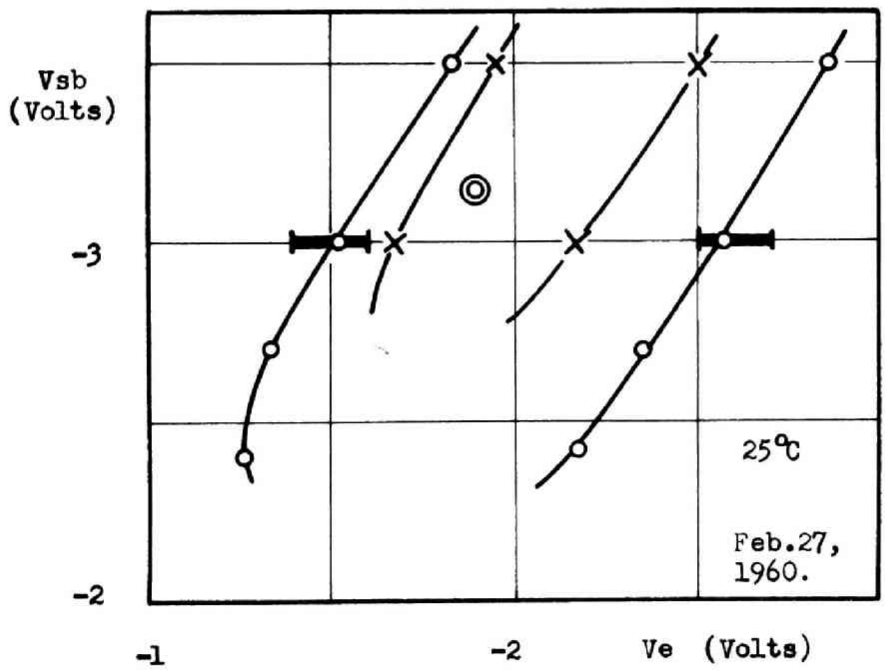


Fig.7.6-Performance margin of the MQ of the KDC-I (four 12-bit ring counters with a decimal adder, a complementer and an add/sub.-1 counter, consisting of 64 basic logic circuits).
○ : when only one pulse recirculates in each register.
× : when the addition of randam numbers is performed.

7 in Fig. 7.7 which have been measured at the Computation Center (this period corresponds to the life time of the serial number 625 magnetic drum).

7.4 Buffer Amplifier

The number of fan-outs required for a basic logic circuit in case of logical design varies considerably; e.g., the output signal of a particular logic circuit used in the timing pulse generator logic circuit should be transmitted to about 100 inputs of other basic logic circuits. For this kind of situation, a buffer amplifies which has a current gain was designed by Mr. Ochiai and Mr. Takase of Hitachi, Ltd. during the time of the logical
8 design of the computer. In Fig. 7.8 the circuit diagram of the buffer amplifier and its logical diagram symbols are illustrated. The buffer amplifier is logically an invertor without a time delay, though actually there is a time delay of approximately 0.1μs which has no effect in the actual operation of logic circuits. It has one high impedance input which accepts either ON or NOT output signal, and has two equivalent outputs, each of which can drive no greater than 10 inputs of basic logic circuits assembled in any parts of any computer racks. The author was not quite willing to use this circuit since the same thing could be done with a much simpler circuit. The author suggested to attach emitter-follower circuits to Type A package, which was not realized at that time. However, it was reported that such a circuit was developed and used in NEAC 2206 computer of Nippon Electric Co.,Ltd.(Ni 1).

7.5 Some Other Circuits

Various electronic circuits were designed, but most of them were rather conventional ones. However, the read amplifier of the photo-electric tape reader (PTR) suggested by the author should be mentioned here. The circuit itself is also a conventional transistorized Schmidt circuit (or voltage comparator), but the application of its special function to the PTR

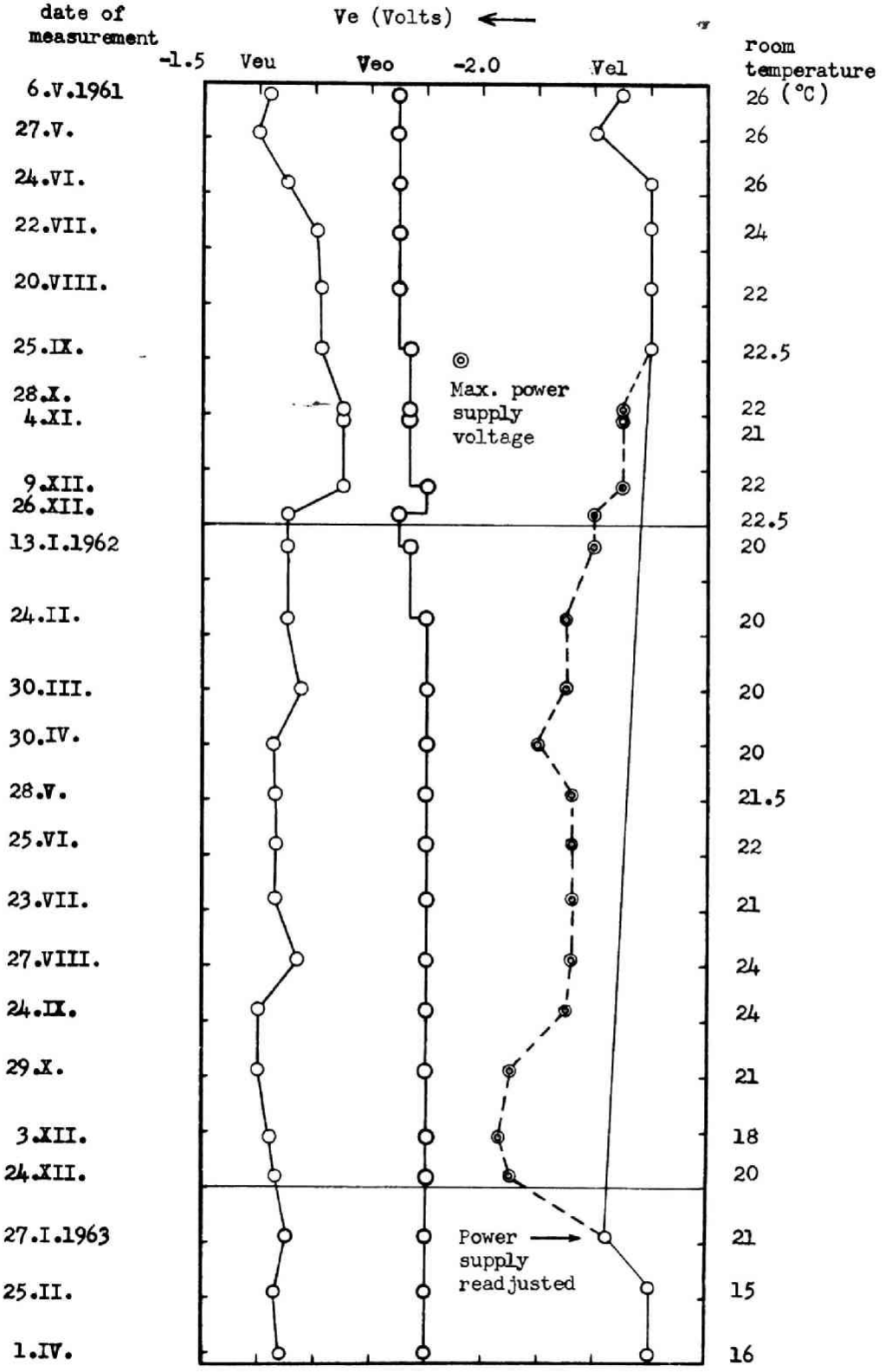


Fig.7.7-Variation of the Ve margin of the KDC-I with time measured while one hundred places of the base of the natural logarithm e were being computed repeatedly.
Ve : Supply voltage for the transistor emitter,
Veo: Set value of the Ve,
Veu: Upper boundary of the Ve margin,
Vel: Lower boundary of the Ve margin.

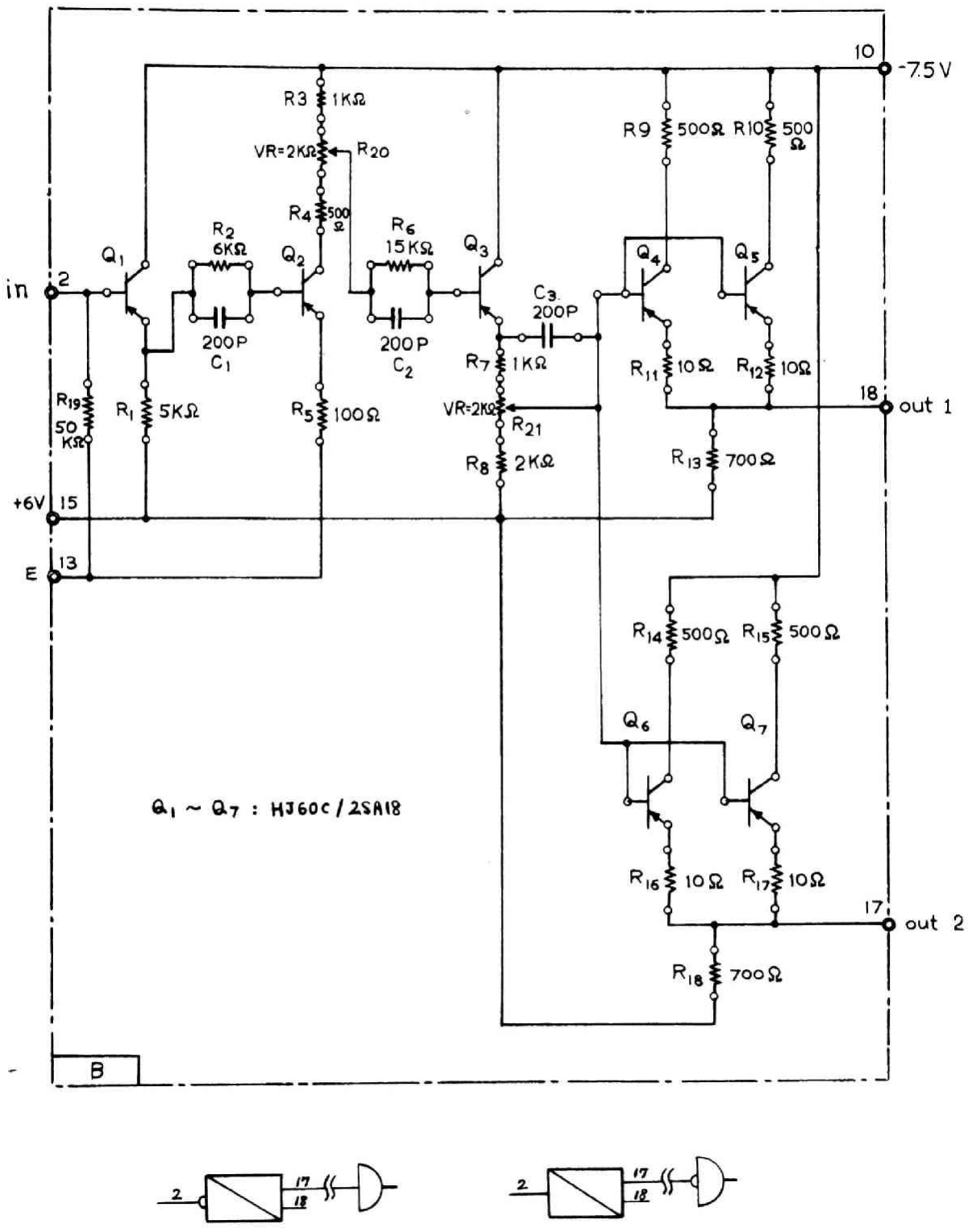


Fig.7.8-Circuit diagram of the buffer amplifier and its logical diagram symbols.

is considered new and interesting.

The PTR reads programs or data at the rate of 200 characters/sec. When the instruction (RIN/14) is executed, 14 characters or less are read by the PTR continuously. The PTR is designed to read even one character by the instruction(RIN/ 1), which puts rather severe operating condition to the PTR, since the information density of paper tape is 4 characters/cm.

The output of the photo-transistor of the PTR which detects sprocket holes is reshaped by the Schmidt circuit, the trailing edge of the output of the Schmidt circuit is used as the information that one character is read. The paper tape should stop as where the detector situates between the neighbouring sprocket holes. It often stops, however, as where the detector situates on the edge of the next sprocket or the output level of the photo-transistor is near the triggering level of the Schmidt circuit. Since there are mechanical vibration, noise, etc., the output of the Schmidt circuit can be turned on and off by them (Chattering), which causes, though rare, undetected errors in read operations.

To solve this problem the backlash or hysteresis characteristic is given between the ON and OFF triggering levels of the Schmidt circuit whose voltage difference is set greater than that of noise by increasing the feedback loop gain of the circuit. By this way the chattering was prevented. The Schmidt circuit with the hystereses characteristic is not new, however, it can be said that this rather peculiar function of the circuit was suitably applied to solve the problem.

Chapter 8

A UNIT TO DISPLAY THE PERFORMANCE MARGIN OF THE BASIC LOGIC CIRCUIT

8.1 Introduction

It should be pointed out that the remarkable progress in the field of electronic digital computers owes greatly to the progress in the field of measuring instruments. Without the high sensitivity and wide frequency band oscilloscope, it would be almost impossible to construct high speed large scale computers. Various instruments based on various novel ideas have been made to date in the field of computers. Among them are the sampling oscilloscopes, memory core testers, core matrix memory plane testers, etc.

A new instrument which was suggested by the arthor was constructed to display the performance margin of the basic logic circuit of the computer. In this chapter details of this unit will be described.

From the aspect of the construction of a digital computer which usually consists of many basic logic circuits, the measurement of the performance margin due to the change of marginal checking parameters of the basic logic circuit is especially important, and is customarily done for all basic logic circuits. Circuits which satisfy a pre-established condition of the performance margin are selected and used.

Point-by-point measurement is very troublesome and moreover inefficient. By using this display unit, it is expected to become much easier

(1) to measure the effect due to variations of values of various parameters such as the width of an input pulse and so on, and

(2) to measure the performance margin rapidly so that all basic logic circuits used for a computer can be easily examined to a satisfactory degree.

Furthermore, by appling the same principle used for this display unit it is expected that

(3) it is possible to construct a performance margin display unit for other types of logic circuits, and

(4) not only the performance margin of a basic logic circuit, but also that of more complex logic circuits consisting of many basic logic circuits can be displayed.

In the following sections, marginal checking parameters of the basic logic circuit, the principle of the display, the organization of the display unit, errors in the measurement, an example of the display and the application of the principle will be presented.

8.2 Marginal Checking Parameters of the Basic Logic Circuit

Concerning the basic logic circuit of the KDC-I, i.e., the transistorized dynamic flip-flop circuit whose functional details were presented in the previous chapter, the supply voltage for the transistor emitter (Ve) and the peak voltage of the strobing pulse (Vsb) are considered to be the two most important marginal checking parameters of the circuit. Thus, in practice, the following measurement is performed. A train of pulses is applied to the input of a basic logic circuit and values of these voltages as marginal checking parameters are varied, then the region of satisfactory operation and the region of unsatisfactory operation are measured by observing the waveform of the output pulse trains on the screen of an oscilloscope. Results obtained are plotted on graphs in which the Ve is taken as abscissa and the Vsb is taken as ordinate.

This Ve-Vsb graph is important, since the operating states of most components of the basic logic circuit, e.g., resistors, condensers, diodes and the transistor, can be learned from it. Thus the graph is used as the basis for the selection of basic logic circuits. Instead of writing the Ve-Vsb graph by plotting data one by one, the graph can be displayed on the screen of a cathode-ray-tube by applying the principle suggested by the author.

8.3 The Principle of the Display

In Fig. 8.1 a block diagram of the display unit is shown. Each functional unit in the block diagram will then be presented in turn.

TC : Basic logic circuit under test.

ST : Basic logic circuit used as a standard for comparison.

PG : Pulse generator which generates a train of pulses which is applied to the input of the two basic logic circuits TC and ST.

CC : Coincidence circuit which tests the coinciodence of a proper sequence of output signal from the TC with that from the ST.

PSE : Power source for the emitter of a transistor of the basic logic circuit whose voltage Ve is periodically varied within a proper voltage range.

SPG : Strobing-pulse generator, the peak voltage (Vsb) of which is periodically varied within a proper voltage range (or modulated) by the output of the MSG described below.

MSG : Modulation signal generator.

CRT : Cathode-ray tube (or used as a cathode-ray tube oscilloscope).

The output voltages of the PSE and the MSG are applied to the horizontal and vertical axis of the CRT (oscilloscope) respectively. The speed of variation of both voltages should be so chosen that it is possible to obtain a suitable raster on the screen of the CRT. In case of the actual display unit, the output of the PSE is the modulated one with a 50c/s sinusoidal wave; and the output of the MSG is a saw-tooth wave whose rate of repetition is 1/2 c/s. Furthermore, by the output of the CC, the beam intensity of the CRT is modulated. In case of the actual display unit, the test of the coincidence is done approximately every 52μs. If coincidence is taken or the TC is operating normally, the spot on the screen of the CRT is illuminated. Otherwise, the spot is not illuminated. The location of the spot on the screen of the CRT is decided by the corresponding voltages of the Ve and Vsb. Thus the region of normal operation of the Ve-Vsb characteristic is illuminated on the screen of the CRT.

Because it is simple to observe the Ve-Vsb characteristic by using

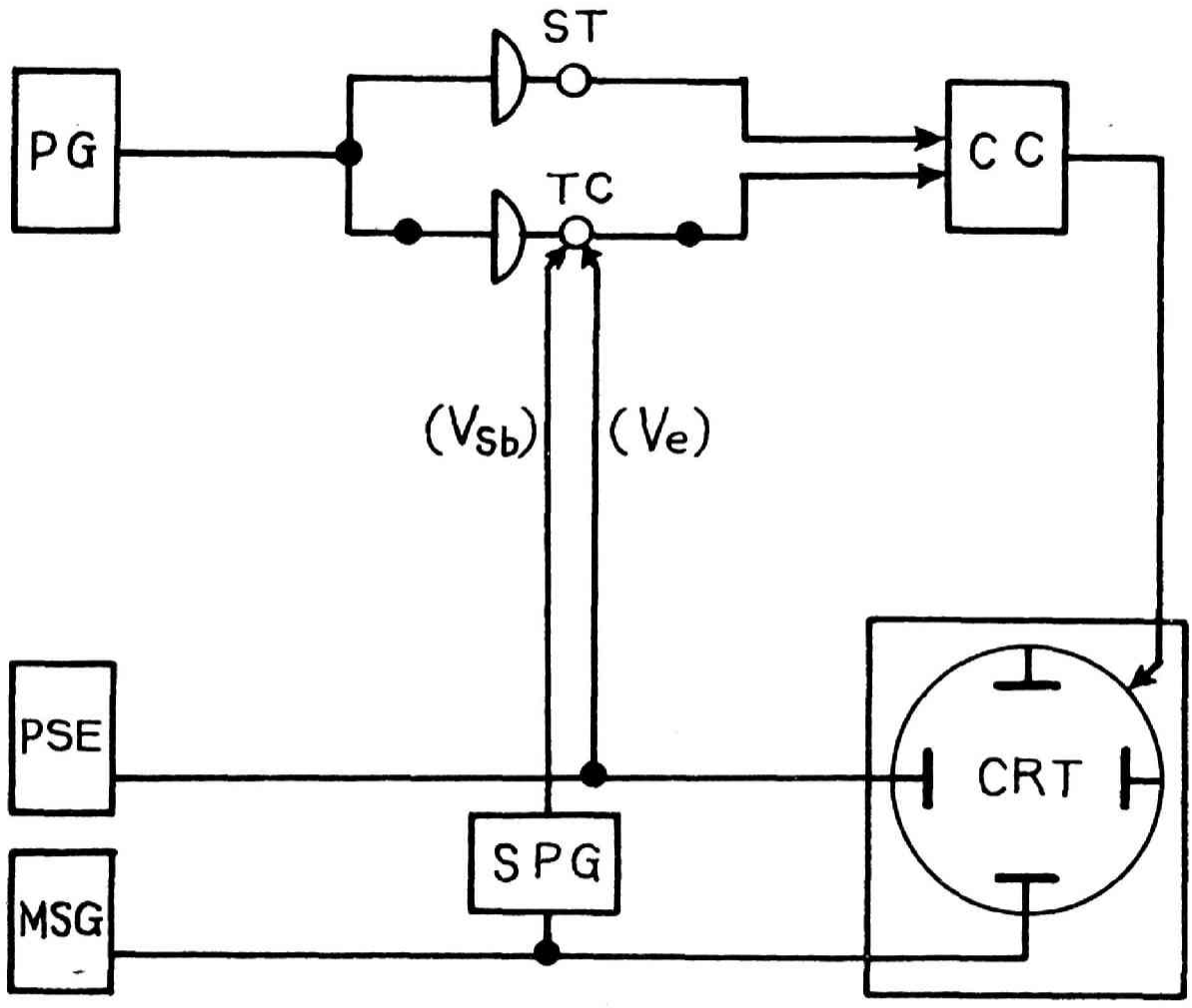


Fig.8.1-Block diagram of the display unit.

this unit, it is expected to become much easier to measure the effect due to variations of various other perameters such as voltage, waveform, operating frequency, the condition of fan-ins and fan-outs, etc. Moreover, the performance margin is slightly affected by the pattern of a train of input pulses in most cases, though it is undesirable. Thus it should be regarded that the pattern of a train of input pulses is also a kind of a parameter to the Ve-Vsb characteristic. In an actual display unit, repetition of a 12-bit pattern is used (which corresponds to the repetition of the same pattern approximately every 5μs, i.e., the one-word time of the computer).

Since the principle of display can be regarded as based on a sort of comparison method, it is possible to display not only the performance margin of a basic circuit, but also that of more complex logic circuits consisting of many basic logic circuits. Moreover, it is important to display the performance margin when the NOT output of a basic circuit is applied to the input of the TC.

8.4 The Organization of the Display Unit

A display unit based on the principle described in the previous section was actually constructed and completed in May 1960. In this section the actual display unit is described in detail.

In Fig. 8.2 a view of the display unit is shown, and in Fig. 8.3 an example of the Ve-Vsb characteristic of the basic logic circuit of the computer is displayed on the screen (diameter : ca. 120mm or ca. 5") of the CRT of the display unit.

The unit is mounted on the 19" standard rack as shown in the figure, and consists of four parts. From top to bottom they are: -

(1) Pulse generators which are used for all circuits of the unit. The repetition frequency and the width of a pulse can be varied.

(2) Display subunit which contains a CRT, modulation circuits, etc.

(3) DC power supply.

(4) Logic circuits which form the CC, PG. etc.

The main features of the display unit are ;-

(1) The Ve and Vsb are varied automatically in the proper voltage range so as to get a proper raster on the screen of the CRT. However, they can be varied manually after setting an anto/manual switch.

(2) The Ve and Vsb can be calibrated on the screen of the CRT.

(3) Other parameters beside Ve and Vsb can be chosen. For example, the number of fan-ins and fan-outs, voltage, wavefrom,frequency, or artificial noise superposed on Ve can be a third parameter.

(4) Check-point terminals are established on important parts of the circuits of the display unit.

(5) The 120mm (5") diameter long persistence CRT is used (since errors originating in the measurement decrease as the speed of the sweep decreases).

(6) With some modification of the display unit, it is possible to measure

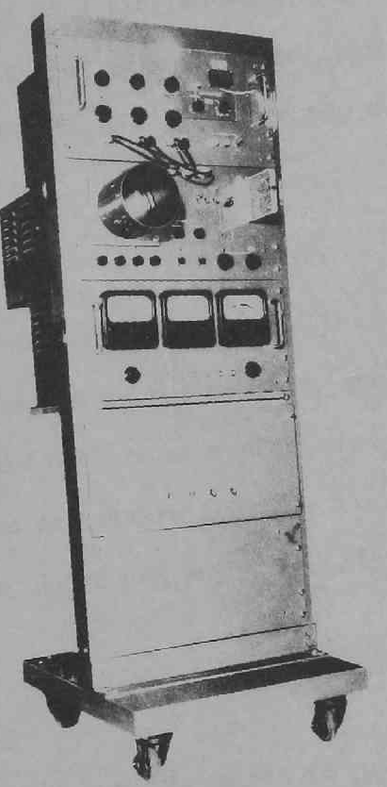


Fig.8.2-View of the display unit.

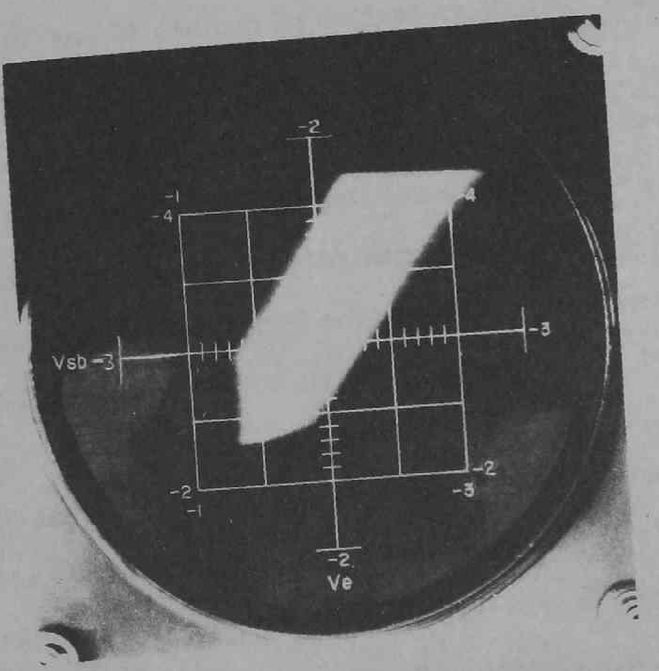


Fig.8.3-An example of the display.

not only the performance margin of a single basic logic circuit but also that of logic circuits consisting of several basic logic circuits.

The details of the main functional units are now described by referring the block diagram of Fig. 8.1.

1. PG : Pulse Generator

A tweleve-bit ring counter composed of basic logic circuits is used. Thus any 12-bit pattern can be set in the counter, and the output from it becomes the repetition of this pattern at approx. every 52μs. For example, the following train of pulses or pattern is used, i.e. 1011111111111. It is obvions that any other train of pulses, e.g., a random pulse train, can be used.

2. CC : Coincidence Circuit

This is a most problematic circuit, since it performs the comparison between the normal output of the standard basic circuit ST and the output of the tested circuit TC under the condition where the normal operating state of the output of the TC can not be defined uniquely. Thus a compromise is made from the practical point of view.

In some cases, the waveform of the output is distorted, and error might occur at the input of the next stage. Thus this can be used as a criterion. I.e., if it is accepted by the normally operating basic logic circuit which is cascaded to the tested one, it should be regarded as a satisfactory output. If it is not, then it should certainly be regarded a dissatisfactory output.

One more point is the interval of the test of coincidence. Apparently it should neither be done at every one-bit time, nor at extremely long intervals. However, it is known from the practice that the test of 10-bit-time interval is satisfactory, since the Ve-Vsb characteristic is scarcely affected by increasing the interval of the test. An interval of 12-bit

time is chosen since it corresponds to 12-bit ring-counter type registers of the computer.

A repetition pulse of very long interval can be generated and the test of very long interval can be performed. This feature is useful when the Ve and Vsb are varied manually.

The actual circuit of the CC is shown in Fig.8.4 together with the TC, the ST and the PG. To is the test pulse of a 12-bit interval or 52μs interval. CC1 and CC2 are coincidence logic circuits which test the coincidence of 11-bit pulse trains of the ST and the TC. If the coincidence is met, CC2 is fired and its output is amplified by CCA, and its output modulates the intensity of the beam of the CRT. Moreover, the CCA performs the blanking during the time interval between Vsb sweeps.

3. PSE : Power Source for Emitter

The sweep of the Ve is performed by 50c/s sinusoidal wave synchronized with the commercial ac line frequency (50 c/s in the Kanto district, 60c/s in the Kansai district, this display unit is now being operated in the Kanto area). The Ve changes in the range (-3V, -1V). The use of the 50c/s sinusoidal wave has the advantage of reducing the effect of ham.

4. SPG and MSG : Strobing Pulse Generator and Modulation Signal Generator

The output of the SPG is the strobing pulse modulated by 1/2 c/s saw-tooth wave. The Vsb changes in the range (-2V, -4V). The Signal is sent to the CCA for the blanking of the CRT during the time interval between Vsb sweeps.

5. A Ve-Vsb raster on the CRT screen

The CRT used is the type 5ABP2, 120mm or 5" medium persistence (low level persistence lasts for a long period) cathode-ray tube.

Since Ve : 50c/s, and Vsb : 1/2 c/s, a raster of about 200 horizontal

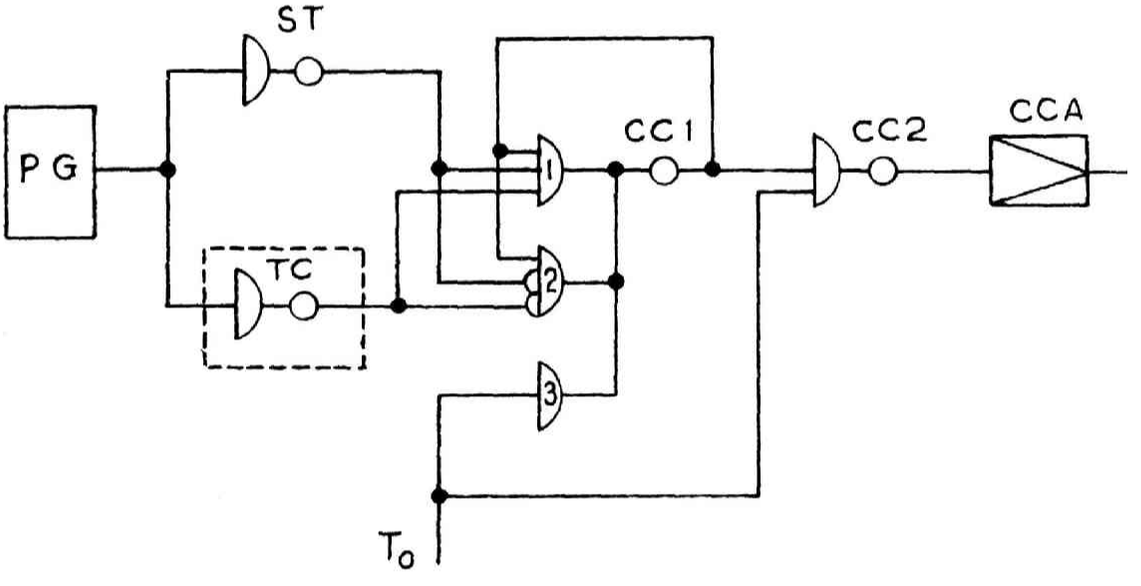


Fig.8.4-Coincidence circuit (CC).

lines is obtained. The sweep of the Vsb is not synchronized.

Because the test is done at every 52μs and time needed for a sweep of a single horizontal line is 10ms, a horizontal line consists of 200 test spots. Thus a raster consist of 200 x 200 test spots.

The deflection factor of the CRT is 2cm/V which is scaled on an illuminating graticule.

8.5 Errors in the Measurement

As is described in previous sections, the principle of measurement is based on a variety of conditions. However, these measuring conditions may be considered appropriate since they are not far from being the actual operating ones.

The apparent error in the measurement is caused by the sweep in relation tothe time interval of the test. The sweep speed of the Vsb is very slow, thus causing no problem. However, the sweep speed of the Ve is not so slow, and its effect can be estimated as follows:

Let $Ve = Vo \sin 2\pi ft + Veo$,

Where Vo = 1 volt, f = 50c/s and Veo = -2v.

Thus, $\Delta Ve = 2\pi f\ Vo \cos 2\pi ft \cdot \Delta t$,

and $Max(\Delta Ve) = 2\pi f\ Vo \cdot \Delta t$,

which corresponds to the center area of the screen of the CRT.

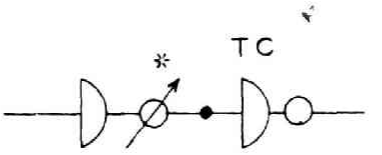
Let $\Delta t = 52\mu s$, then $\Delta Ve \cong 16mV$.

The beam sweeps from left to right and back on its return. Thus near the center of the screen of the CRT there is an error of the order $2\Delta Ve \cong 32mV$.

The deflection factor of the CRT is 2cm/V, and 100mV scale is given to the graticule. Thus the error causes no serions problem. However, there is a slightly larger error than is estimated at the periphery of the normally oprating area. No exact reason is found for this phenomena. However, it is assumed that there might be a hysteresis in the operation of the basic logic circuit with regard to the variation of the Ve.

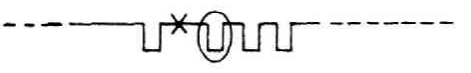
8.6 An Example of Measurement

Marginal checking parameters of the circuit is the Ve and the Vsb. Fig. 8.5 shows an example of measurement when the width of an input pulse is taken as a third parameter.



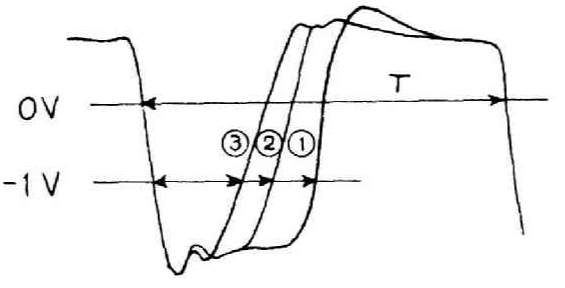
* The pulse width is changed in this circuit.

Ambient temperature: 23.5°C.



A train of input pulses: repetition of the following 12-bit pattern:- 101111111111.

Test interval : 12-bit time or 52 μs.



Waveform of an input pulse.
T : 4.35 μs,(230kc/s).
Pulse width,
1 : 2.05 μs,
2 : 1.45 μs,
3 : 1.10 μs.

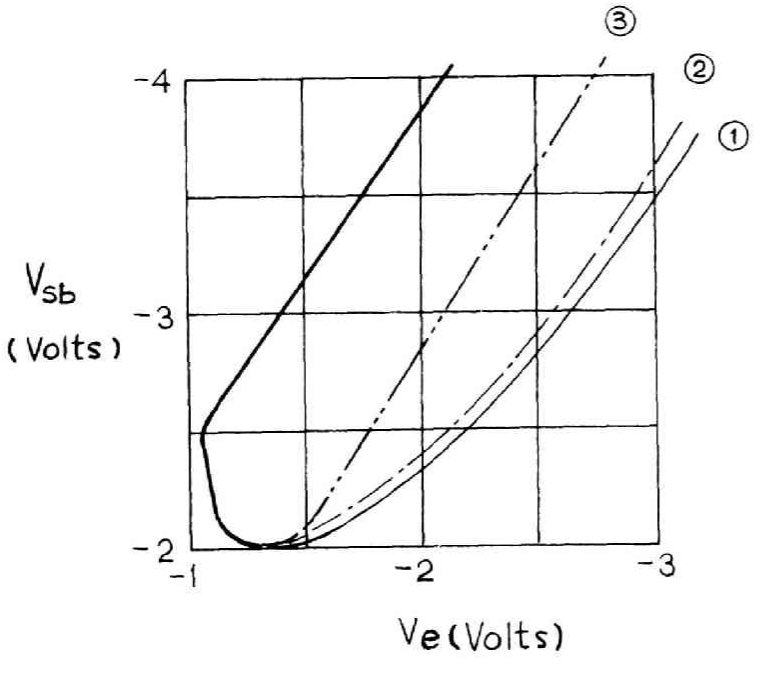


Fig.8.5-Ve-Vsb characteristic with input pulse width taken as a parameter.

8.7 The Application of the Principle

Some examples of the application of this principle are presented by showing schematic diagrams. Fig.8.6 shows the method of measurement when a NOT output is applied to the input of the circuits. When the coupling between a read amplifier of a magnetic drum and a basic circuit is a matter of concern, the method shown in Fig.8.7 can be utilized. In Fig.8.8 the method of measurement of the Ve-Vsb characteristic of an adder is shown.

8.8 Conclusion

The details of the display unit have been described. Various interesting applications of this display unit are possible. The principle of measurement can be applied not only to the dynamic type circuit, but also to other types of logic circuit. For example, in case of a parametron logic circuit, the dc bias current and the exciting voltage can be taken as two checking parameters, and a performance margin of a parametron can be displayed in the same way.

It was reported in January 1962 that a display unit for a static logic circuit based on this principle was constructed at Osaka University (Ha 1).

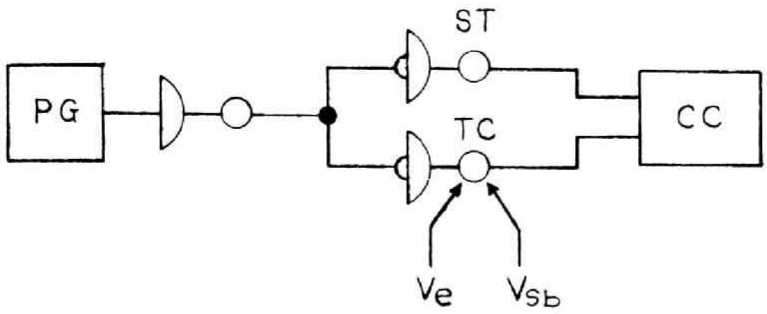


Fig.8.6-Measurement of the performance margin when a NOT output is applied to the input.

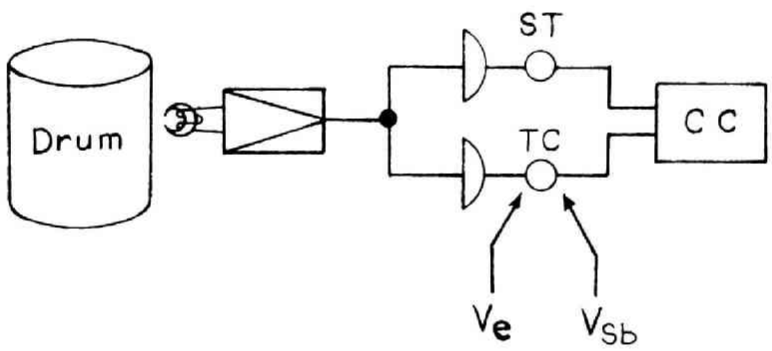


Fig.8.7-Measurement of the performance margin when the output of a read amplifier of a magnetic drum is applied to the input.

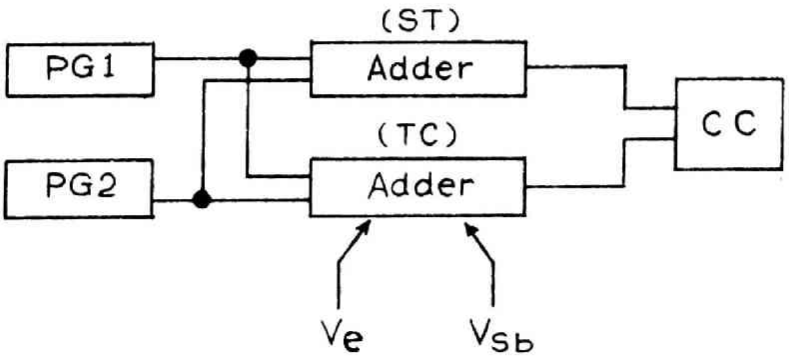


Fig.8.8-Measurement of the performance margin of an adder.

Chapter 9

CONCLUSION

In concluding the thesis the main points of the results of the studies are presented.

(chapter 2)

1. An electronic digital computer, the MDC-I, was successfully completed, which satisfies the specified functions and can be concluded to be a fairly reliable computer from actual operation.

   The installation of the MDC-I at the University was at the beginning of August 1960. After spending one month for the reassembly and for various tests, the MDC-I has been utilized and maintained at the University. Various interesting applications have been reported to date.

   The machine is functioning about 90 % of the total time that it is turned on in spite of insufficient maintenance personnel.

   From our experience of its operation, the main causes of difficulty were in the mechanical defects in the input and output components of the machine. The most troublesome part was incomplete soldering at the terminals of the back panel logics wiring. Several transistors have broken down. Most of these breakdowns were due to incomplete evaluation of their power dissipation. These circuits have now been remodeled and no trouble has been registered from these circuits since then. However, some of the faulty transistors showed unfavorable changes of characteristics due to unknown reasons. No diode trouble has yet been recorded to date in spite of initial anxiety. However, the speed of the computation seems progressively slower as the amount of information to be processed increases.

(chapter 3)

2. By providing a rather rich repertory of instructions and a decimal structure to the machine, the programming for the computer (with machine or symbolic codes) is comparatively easier than for other machines.

3. Main features of the instructions are listed in section 3.3 of chapter 3, which are not repeated here.

4. Instructions for logical operations were devised; thus it became possible that not only operations between words or between decimal digits, but also operations between numbers of different weights and between bits can be performed.

5. Instructions for special operations are new types of instruction which have interesting properties.

(chapter 4)

6. A magnetic tape system was successfully completed, which satisfies the specified functions, in spite of the difficulties at the time of the design.

7. There are about fifteen more items which should be described here, but they are presented in section 4.7 of chapter 4.

(chapter 5)

8. The simplest form of reading the initial read-in routine is shown with a proof.

(chapter 6)

9. The logical design had been conducted under rather severe restrictions on the available number of fan-ins and fan-outs of the basic logic circuits. (A portion of its effect can be found in Fig.6.8)

10. A method was devised to obtain the simplest normal function with a restriction on the operation of an input gate in which no more than a single AND gate can send binary 1 at the same time.

11. Arithmetic elements were designed under the above restriction; it was

shown that in all examples the number of terms never increased, and only a slight increase of the number of diodes is caused.

12. Steps of the logical design and main design features are not repeated here ; they are found in section 6.6 of chapter 6.

(chapter 7)

13. Some modification to the ETL type transistorized dynamic circuit was made to improve its characteristics by Professor T.Sakai and Professor H.Hagiwara.

14. A procedure was set up in which the performance margin of all basic logic circuits is measured, and then that of several functional blocks is also measured. By applying this procedure the adjustment of the computer is likely to become much simpler (The thorough examination of the results of logical design and the wiring of the computer is required).

15. The emitter voltage margin of the whole computer fructuateswith time, however, it can be concluded it is very stable.

16. The hysteresis characteristic of the Schmidt circuit was suitably applied to solve the input circuit chatter problem of the photo-electric tape reader.

17. It can be concluded that a transistor circuit is very stable, if it is properly used.

(chapter 8)

18. A new unit to display the performance margin of the basic logic circuit was devised and completed. It was reported in January 1962 that a display unit for a static logic circuit based on this principle was constructed at Osaka University.

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitudes to Professor Ken-ichi Maeda for his guidance and support during the course of this investigation.

He is also very much obliged especially to Professor Takeshi Kiyono, Professor H.Nishihara, Professor Toshiyuki Sakai, Professor H.Hagiwara and Assistant Professor Susumu Kato for their guidance and valuable suggestions.

He owes also a considerable debt of gratitude to the Director of the Kanagawa Works Dr. S.Takada, the director of the computer section Mr. K. Iwama, Mr. Y.Hatano, Dr.K.Furuya, Mr.E.Ota, Mr.S.Iyobe, Mr.H.Mandai, Mr. H.Yazaki and Mr.K.Nishi of Hitachi, Ltd. for their co-operation in the design, construction and adjustment of the computer.

His hearty thanks go to Assistant Professor H.Matsumoto of Kobe University , Assistant Professor I.Kimura and Mr.M.Nagao of Kyoto University, the then graduate students of Kyoto University Mr.H.Fujimoto and Mr.K. Hashimoto for their kind co-operation in the development of the basic logic circuit of the computer, and to Mr.H.Nishio and Mr.K.Takao of Kyoto University, and the then graduate students Mr.T.Fukinuki and again Mr.K.Hashimoto for pointing out errors in the logical design of the computer.

He is very much obliged to Mr.Y.Shibatani of Kobe Industrial Co., and Mr.I.Yagi of Kyoto University for their efforts in the development, inspection of various electronic circuits, and the maintenance of the KDC-I.

Great appreciation is expressed by the author to all of the others who have contributed very substantially to the successful completion of the KDC-I.

REFERENCES

(Al 1) G.M. Altman, L.A. DeCampo and C.R. Marburton, "Automation of Computer Panel Wiring," AIEE Transactions Paper 60-128, January, 1960.

(El 1) R.D. ELKOFN and A.P. MITT, "Dynamic Circuit Techniques Used in SEAC and DYSEAC," Proc. I.R.E., vol. 41, pp.1380-1387; October 1953, and also "Computer Developments (SEAC and DYSEAC) at the National Bureau of Standards", NBS Circular 551, January 25, 1955.

(Fe 1) J.H. Felker, "The transistor as a digital computer element," Review of Electronic Digital Computers, A.I.E.E., Special Publication S-44, p. 105-109; February, 1952.

(Gr 1) E.M. Grable et al., "Handbook of Automation, Computation, and Control Vol.2," John Wiley & Sons, Inc., New York, 1959.

(Ha 1) A. Hashimoto et al., "A Test Unit of a Packaged Circuit," ECTC* of the IECEJ**, January 19, 1962.

(Ka 1) M. Karnaugh, "The Map Method for synthesis of combinational logic circuits," Trans. Am. Inst. Elec. Engrs., Pt. 1. Vol. 72, pp 539-598 (1953).

(Ki 1) Takeshi Kiyono; "SYCS-1, the Symbolic Coding System of the KDC-I," Electronic Computer Tech. Committee, IECEJ**, Jan. 16. 1961.

(Ki 2) T. Kiyono, "Utilizing Magnetic Tape Units of KDC-I," KDC-I Report PT-001 1962/002, Kyoto University Computation Center.

(Kr 1) J.B. Kruskal Jr., " On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," Proc. Am. Math. Soc., vol7, pp. 48-50, 1956.

(Kr 2) D.B. Kirby and C.M. Rosenthal, "Computer Program for Preparing Wiring," Communication and Electronics, AIEE, November, 1961 No.57 pp 509 513.

(Ku 1-3) KDC-I Manuals; Kyoto University Computation Center,
Vol. 1 : "Programmer's Manual of Operation".
Vol. 2 : "Programmer's Reference Manual".
Vol. 3 : "Programmer's Manual of Library".

(Ku 4) "Magnetic Tape Test Programs" ; Kyoto University Programming Group, Kyoto University Computation Center.

(Le 1) R. Ledley, "Digital Computer and Control Engineering," McGraw-Hill Book Co., Inc., 1960.

(Lo 1) H. Loberman and A. Weinberger, "Formal Procedures for Connecting Terminals with a Minimum Total Wire Length," J. Assoc. Computing Machinery, vol. 4, no. 4, pp. 428-437, October, 1957.

(Mi 1) Y. Miyagi, "NEAC 2206," ECTC* of the IECEJ**, September 27, 1962.

(Na 1) Makoto Nagao, "Construction of an Automatic Programming System Based on ALGOL 60," Master Thesis at Kyoto University, 1961.

(Ni 1) H. Nishino, Takahashi, Matsuzaki, Aiso, Kondo and Yoneta, "A Transistor Computer Denshi Mark IV," The Journal of the IECEJ**, pp.1039-1045, Nov. 1959, and also ECTC* of the IECEJ**, April 25, 1957 and February 20, 1958.

(Ph 1) M. Phister, Jr., "Logical Design of Digital Computers," John Wiley & Sons, Inc., New York, 1958.

(Pr 1) A.I. Pressman, "Design of Transistorized Circuits for Digital Computers," John Wiley & Sons, Inc., New York, 1959, pp.254-257.

(Qu 1) W.V. Quine, "The Problem of Simplifying Truth Functions," American Mathematical Monthly, 59, 521-531 (1952).

(Qu 2) W.V. Quine, "A Way to Simplify Truth Functions," American Mathematical Monthly, 62, 627-631 (1955).

(We 1) E.W. Veitch, "A Chart Method for Simplifying Truth Functions," Proc. Assoc. Computing Machinery, pp 127-132, May 2-3, 1952.

Articles on the KDC-I and the related subjects reported by the author are listed below;

(Ya 1) K. Maeda, T. Sakai, S. Yajima, et al., "On the Kyoto University Digital Computer KDC-I," ECTC* of the IECEJ** March 23, 1961.

(Ya 2) T. Kiyono, T. Sakai and S. Yajima, "On the Design of the Kyoto University Digital Computer KDC-I," Memoirs of the Faculty of Engineering, Kyoto University, Vol. XXV, Part 1, pp 38-62, January, 1963.

(Ya 3) K. Maeda, T. Sakai, S. Yajima et al., "The Magnetic Tape System of the KDC-I," ECTC* of the IECEJ**, August 24, 1961.

(Ya 4) S. Yajima, "On the Block Number of the Magnetic Tape," Record of the 1961 Convention of the Information Processing Society of Japan.

(Ya 5) T. Kiyono, T. Sakai and S. Yajima, "The Magnetic Tape System of the Kyoto University Digital Computer," to be published in the Memoirs of the Faculty of Engineering, Kyoto University.

(Ya 6) S. Yajima, E. Ota, S. Iyobe, "On the Digital Computer HITAC 102D," Hitachi Review, pp 1-5, December, 1960.

(Ya 7) S. Yajima, "A Unit to Display the Performance Margin of a Dynamic Flip-flop Circuit," ECTC* of the IECEJ** January 16, 1961.

(Ya 8) S. Yajima et al., "A Unit to Display the Performance Margin of Basic Logic Circuit," Record of the 1960 Autumn Convention of the IECEJ**.

(Ya 9) S. Yajima and S. Takada, "The Wiring Design of Logic Circuits by a Computer," Record of the 1960 Spring Joint Convention of the IECEJ** and others.

The brief reports on the KDC-I are found in the following articles;

(Ya 10) S. Yajima. "Outline of the KDC-I," Record of the 1960 Autumn Convention of the Kansai chapter of the IECEJ**.

(Ya 11) S. Yajima and . Iyobe, "Outline of the System of the KDC-I and its Logical Design," Record of the 1960 Convention of the Information Processing Society of Japan.

(Ya 12) T. Sakai, S. Yajima and S. Iyobe "The System and Logical Design of the KDC-I," Record of the 1960 Autumn Convention of the IECEJ**, No.78.

(Ya 13) T. Kiyono, T. Hagiwara, S. Yajima, et al., "On the Instruction System of the KDC-I," ibid., No.79.

(Ya 14) T. Sakai and S. Yajima, "On the Logical Operation of the KDC-I," ibid., No.80.

*ECTC : Electronic Computer Technical Committee

**IECEJ : Institute of the Electrical Communication Engineers of Japan

Appendix A

(1)

THE CONSOLE OF THE COMPUTER

A program tape is placed in the Input Unit (usually a photo-electric tape reader), and the operator sets and/or depresses suitable buttons on the operator's console. Then the computation (the read-in and the computation of the program) begins and results are printed via the output components.

The operator's console or simply console is used for the supervision of the computation, the machine check and adjustment. Buttons (or switches) and lamps which are used for the supervision are placed in the front panel of the console. The drawing of the front panel of the operator's console is shown in Fig.A . At the rear side of the console, the switches which are mainly used for the machine check and adjustment are placed.

The functions of each lamps and buttons on the panel are described in turn mainly from left to right part of the panel.

If the contents of a bit is one, the corresponding lamp is turned on and if zero, off.

(1) The display of the contents of operation registers and counters.

The array of lamps under the name REGISTER is used. The first row of the array has the weight 1 of the BCD. The second has 2, the third has 4 and the fourth has 8. A register or a counter whose contents should be displayed is specified by setting the button which has the name of the register or the counter.

N.B. The c(LC) is displayed in the positions 9 – v, the c(IR1) in the positions 1 - 4, the c(IR2) in the positions 5 - 8, and the c(IR3) in the positions 9 - v. The LA overflow bit and the LA sign are displayed in the weight 1 and 8 of the position v regardless of the R-indicator state.

(2) The selection of the Input and the Output components.

The selection is performed by the instruction SEL, but it is also possible to select them by depressing the buttons which have the

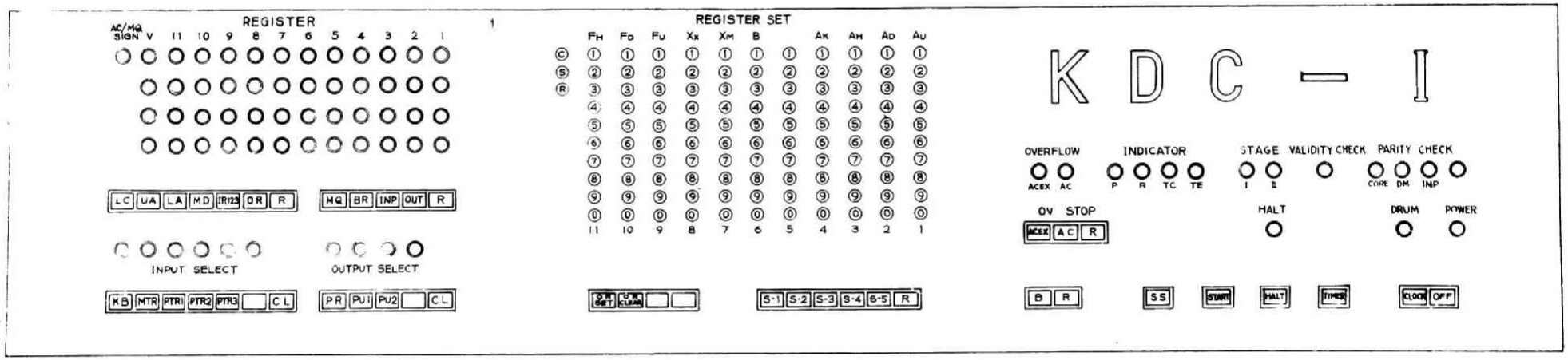


Fig. - Front Panel of the Operator's Console

corresponding names. If selected, the corresponding lamps are turned on.

(3) The manual execution of an instruction at the console.

An instruction word is prepared by setting the REGISTER SET buttons in the middle of the panel of the console. Before setting an instruction in the OR, the computer must be in the state of HALT (confirm that the HALT lamp is on). Then depress the OR CLEAR button and next depress the OR SET button, by this way the instruction can be set in the OR. If the SS button at the down right of the panel is then depressed, the execution of the instruction takes place. And if the START button is depressed, the computer takes the next instruction in sequence. So by this procedure the insertion of an instruction into a program becomes possible.

(4) The JSW Switches.

There are five switches S-1, ..., S-5. Thier function is described in the instruction JSW.

(5) The Break Point Switch.

See the Break Point Part of the Computer Instructions.

(6) The AC overflow and the AC exponent overflow.

If the v position of the AC has a non-zero number in any case, the AC OVERFLOW lamp is turned on. If the v position of the AC has a number other than 0 and 1 in any case, the ACEX OVERFLOW lamp is turned on.

If the AC (or ACEX) OVERFLOW lamp is on as a result of the fixed (or floating) point operation, the computation stops if the AC(or ACEX) OV STOP button is set on. But details are described in the Computer Instructions.

(7) The indicators

The P lamp indicates that a P-type instruction (PSX, SCT, TLU) is being executed. The R lamp indicates that a remainder is obtained by the division (DVJ, FDJ). The TC lamp indicates that the dual parity error of the magnetic tape is checked. The TE lamp indicates that the tape end of the magnetic tape of the currently selected magnetic tape handler is being detected.

Details of each are described in the Computer Instructions.

(8) The operation of the computer at the console.

The Power Switches for the drum and for the computer placed at the front part but not on the panel of the console are set on (and the lamps DRUM and POWER are on), the CLOCK button is then set on (the clock pulse is supplied to the computer), and next the TIMER is depressed (all timing in the computer is synchronized and generated). Then the computer is ready to perform computations.

At this time, all registers and counters,indicators and so on are reset to all zeros except the non-volatile memory (the N-bands of the drum, the core memory, the magnetic tape, while the Q-band of the drum are volatile, therefore reset to zeros) (Non-Volatile memory; Even if the power to the computer is turned off, the contents of the memory are retained and can be used when the power is turned on again.).

It is also possible to execute an instruction manually by the way described in (3).

The START button, if depressed, starts programs from the location c(LC). The SS button, if depressed, advances only one step of a program or an instruction in the location c(LC) is computed.

The HALT button, if depressed, halts the operation of the current instruction at the end of its operation, and turns on the HALT lamp. The HALT lamp is turned on whenever the computation is stopped by depressing the HALT button, by some particular instructions, and by detecting parity and/or validity errors.

The STAGE I lamp is turned on during the instruction time, while the STAGE II lamp is turned on during the execution time.

(9) The check lamps

If a validity error is checked, the VALIDITY CHECK lamp is turned on. If a parity error is checked at the core or the drum memory or at the input

register, the CORE, DM or INP PARITY CHECK lamp is turned on respectively.

(10) The TE button

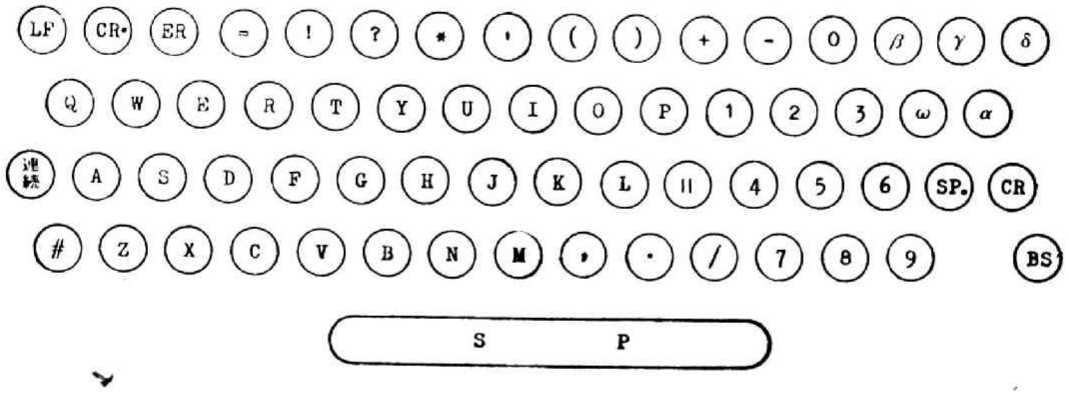
The TE (Tape End) indicator is turned on during this TE button is being depressed. Normally the TE indicator is on at the physical end of the magnetic tape. This button is mainly used for the instruction BLS.

Appendix A

(2)

Table I KDC-I I/O Characters

(1) Keys of the Console Typewriter



(2) Conversion Table of the Characters

CHARACTER (KEY)	ALPHANUM. MODE	NUMER. MODE	CHARACTER (KEY)	ALPHANUM. MODE	NUMER. MODE	CHARACTER (KEY)	ALPHANUM. MODE	NUMER. MODE
0	00	0	D	21	5	Y	42	0
1	01	1	E	22	6	Z	43	1
2	02	2	F	23	7	?	44	2
3	03	3	G	24	8	!	45	3
4	04	4	H	25	9	*	46	4
5	05	5	I	26	0	ER	47	5
6	06	6	J	27	1	NE	48	0
7	07	7	K	28	2	SP·	49	1
8	08	8	L	29	3	CR·	50	2
9	09	9	M	30	4	‖	51	3
+	10	0	N	31	5	≠	52	4
−	11	1	O	32	0	=	53	5
LF	12	2	P	33	1	.	54	6
SP	13	3	Q	34	2	/	55	7
CR	14	4	R	35	3	(	56	8
(ER)	15	5	S	36	4	)	57	9
,	16	0	T	37	5	ω	58	0
'	17	1	U	38	6	α	59	1
A	18	2	V	39	7	β	60	2
B	19	3	W	40	8	γ	61	3
C	20	4	X	41	9	δ	62	4

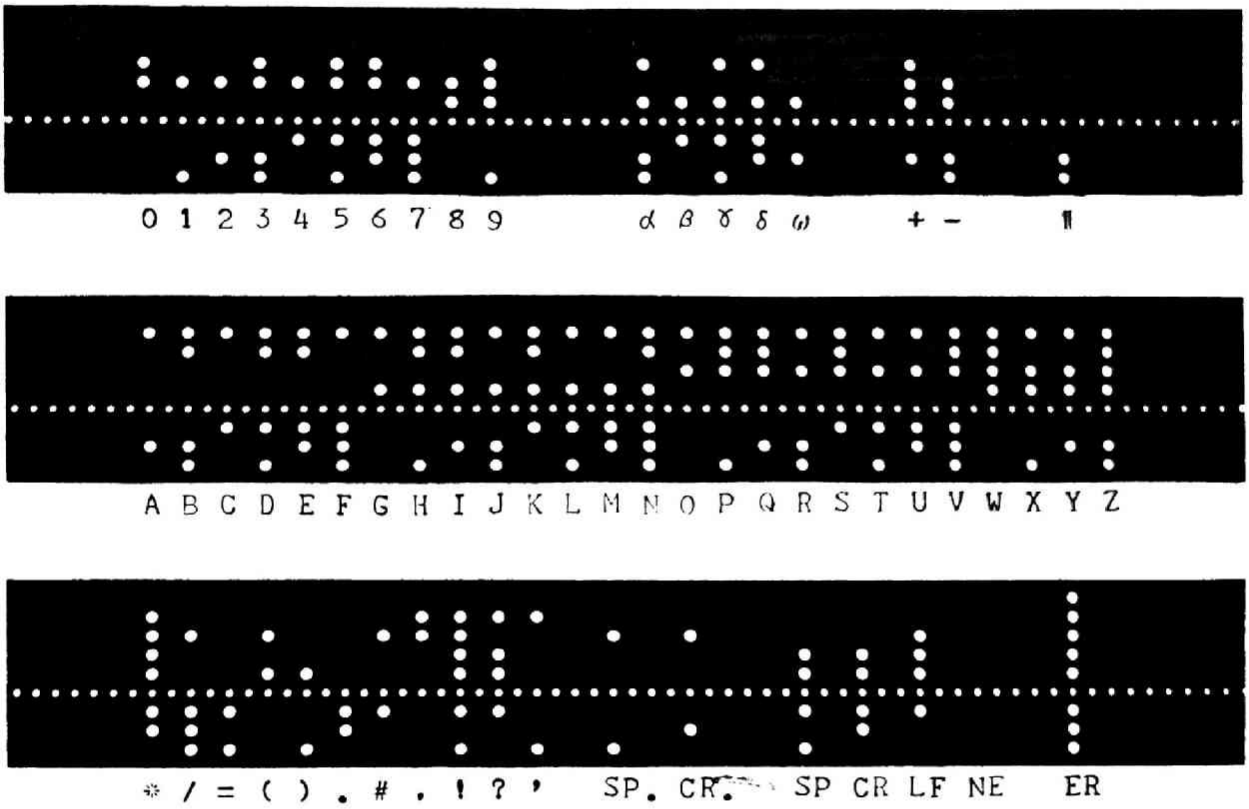
LF (12): "line feed", ignored by tape reader, SP (13): "space", ignored by tape reader, CR (14): "carriage return and line feed", ignored by tape reader.
(ER) (15): "erase" for 6-unit tape, ER (47): "erase", NE (48): "sprocket hole or space", ignored by tape reader, SP· (49): "space", **CR·** (50): "carriage return and line feed".

Appendix A

(3)

Table II Paper Tape Character Coding

Excess-16 Binary Coded Characters



o-weight 64
o- 32
o-even parity bit
o- 16
o- 8
o- sprocket
o- 4
o- 2
o- 1

Appendix B

DEFINITIONS OF THE KDC-I INSTRUCTIONS

Symbols and abbreviations used in this Appendix can be found in chapter 3. Definitions of all instructions are given except that of Logical, Special and Magnetic Tape Operations which are described in chapter 3 and 4 respectively. However, only their tittles are given.

In this Appendix ;-

B.1 Time Required for Computation

B.2 Illegal Instructions

B.3 Illegal Addresses

B.4 Some Notes on Computer Instructions

B.5 The Definition of the Operation of Each Computer Instruction

B.1 Time Required for Computation

The time required for the computation of a program depends on time required for the computation of an instruction, and the sequence of instructions computed in a program. In the description of each instruction, the time required for the computation of each instruction is described. So the programmer can estimate the time required for the computation of his program.

Since an instruction is read from a Storage Register, and executed, the time for the read or the instruction time and the execution time are required for the computation of one step of a program.

The instruction time is decided by the access time of the Storage Register from which the specified instruction is read, and is denoted as Ai.

The execution time is described in two ways according to the type of the operation of each instruction.

(1) In case where an instruction refers at least to the Storage Register specified by the address part of the instruction, the access time, denoted as a data access time Aa, is necessary in addition to the operation time of the instruction, denoted as T.

Thus (T + Ai + Aa) is necessary for the computation of an instruction of this type.

(2) In case other than the above, only the operation time of an instruction is necessary for the execution of the instruction.

Thus (T + Ai) is necessary for the computation of an instruction of this type.

The time required for the computation of each instruction is described in each of the computer instructions. Since the instruction system is the one and a half address system, the average access time can not be used always for the assumption of Ai and Aa.

Rigorous estimation of the computation time is tedious and moreover not so important in most cases, so the assumption is roughly made in the

following way (the average of the longest and the shortest computation time of an instruction is described, TLU and the Input/Output and Magnetic Tape Instructions are excluded);-

(i) Both an instruction and a data in N-bands of the drum memory;-

Multiplication and division instructions - ca. 15ms; others - ca. 10ms.

(ii) Both in Q-bands of the drum memory;-

Multiplication and division instructions - ca. 8.75ms; FAD, FSB, FAA, FSA - ca. 3.75ms; others ca. 2.5ms.

(iii) Both in the core memory: T + Ai + Aa or T + Ai

(iv) An instruction in a Q-band, and a data in a N-band;-

Multiplication and division instructions - ca. 12.5ms; others which need data access time - ca. 7.5ms.

(v) An instruction in the core memory, and a data in a N-band, Aa - ca. 5ms.

(vi) An instruction in the core memory, and a data in the Q-band, Aa - ca. 1.25ms.

N.B. The computation time is unchanged whether or not the modification of the address of an instruction takes place. The time for the modification is included in the operation time T.

B.2 Illegal Instructions

Every instruction has the corresponding 3-digit numerical code, and the code is decoded and the instruction is executed. If a 3-digit number other than the above codes is read for the decoding, which is called an illegal instruction, then the following situation occurs in case by case fashion (the details are not described, since it is not so important);-

(1) The computation stops at the beginning of the execution time in most cases, since the decoding becomes impossible (e.g. 000).

(2) The computation stops after doing some erraneous operations.

(3) The computation does not stop, but the illegal instruction functions as if it is the instruction NOP.

(4) The computation does not stop, but the illegal instruction causes some errors to the operation of some other instructions.

So the Illegal Instruction should not be used.

B.3 Illegal Addresses

Since the address part of an instruction has four digits, and since the address of the SRs are from 0000 to 4249, any number in the range 4250 - 9999 is an Illegal Address when it is used to refer the SR.

The Illegal Address works as if there is a phantom memory whose contents are always all zeros. So in case of reading, zeroes are read from it. And in case of attempting to write, the attemp is made in vain. Thus the computation is not stopped only by this fact.

An error in programming often calls an instruction from the illegal address, and then the illegal instruction whose function part is 000 is read and the computation stops at the beginning of the execution time.

The illegal address or a phantom memory could be used for various ways, but it is highly recommended that it should not be used, because the extension of the memory capacity might be realized.

B.4 Some Notes on Computer Instructions

Some notes described here are often useful in case of programming.

1. Switches and Indicators

There are some switches and indicators which relate to the operation of some instructions. The states of these switches and indicators are equally as important as the contents of registers and counters. As the contents of registers and counters can be displayed on the panel of the console, the states of these switches and indicators are also supervized at the console.

Details of each are described in each of the computer instructions, and general information of each is given in the description of the console. Beside these, there are some busy indicators, which are described in the Magnetic Tape Operations.

2. Instructions Commonly Used for Both Fixed and Floating Point Numbers

The Fixed Point Operations are operated on fixed point numbers, and the Floating Point Operations are operated on floating point numbers. However most of the other instructions are used commonly for both fixed point and floating point numbers. For example, the Word Transfer Operations and Block Transfer Operations are used commonly for both type of numbers. Even the instructions CMP and TLU can be used commonly.

Caution should be taken in case the transfer of a word to the UA. Since the instruction ADD/ does not transfer a number in the overflow bit of a word, and the instruction FAD/ includes the normalization operation. The instruction FSL is used to store the lower half of the double length mantissa of a floating point number in the AC (i.e. the LA part), while the instruction SLA is used to store the c(LA), i.e. the lower half of the double length fraction of a fixed point number in the AC, or a fixed or floating point form remainder in the LA, depending on the condition of the R-indicator.

3. The Contents of Registers after Reading Operations

The contents of registers and counters, and the states of indicators and switches are unchanged by the operation of an instruction unless otherwise described in the computer instruction.

Moreover the contents of any register are unchanged as a result of reading the contents in the operation of an instruction.

4. The Unnormalized Form of a Floating Point Number

In most of the Floating Point Operations, the normalization operation is included. All results by the Floating Point Operations are given in the normalized form except the remainder by the instruction FDJ, if numbers concerned are given in the normalized form. Details are described in the description of each instruction.

5. +0 and -0

Since a register has a sign bit, two states of zero +0 and -0 exist. Either of two zeros will appear in a register. In most cases, zero takes the form of +0. For example, in the operation of addition or subtraction instructions, and the Clear Operation of the AC. Details are given in the description of each instruction.

The +0 and -0 are treated as a same zero in most of the instructions, but some of the Sign Operations, the Control Operations examine only the sign of a register, and moreover the instruction CMP regards that +0 > -0.

B.5 The Definition of the Operation of Each Computer Instruction

In this section the Clear operation and the operation of each of all 95 computer instructions are defined.

Each instruction is classified for the convenience of programmers into each of fifteen categories according to the type of operations performed, and each is described in turn according to the classification and not necessarily in the ascending order of the numerical operation codes. Most of the terms used in this section are defined in the previous sections.

The definition of the operation of each instruction is made in the following way;-

The numerical operation code, the symbolic representation of the instruction, the official name of the instruction, and the computation time of the instruction are written in the first place. Then the operation of the instruction is written by using symbols. Next the details of the operation is given, in which the way of execution, registers and counters referred, and indicators and switches affected are mainly described.

Several notes are added if necessary. A few examples are given if it is convenient to clarify the operation.

1. The "Clear" Operation

A decimal code of any instruction has an even number except that of FMP/ (221) and FMC/ (223). By adding one to any even number code (hence becomes an odd number code), it is possible to clear the c(AC) just before the operation of the instruction of an even number code. The symbolic expression for "Clear" is the slach "/" attached to the last of the three character symbolic code, even though the "Clear" is executed just before the operation of the instruction of an even number.

The precise operation of the "Clear" is as follows;

The c(AC) are replaced by plus zero (The 12-digit c(UA), the 11-digit c(LA) and the AC sign are set off).

The R-indicator (Remainder indicator), the LA sign and the LA overflow bit are set off. (only DVJ and FDJ can set on the above. This "Clear" operation, multiplication and division operations set off the R-indicator. The SLA is only affected by the condition of the R-indicator.)

No additional operation time is needed to that of the instruction of an even number.

e.g. 100 ADD "Add"

101 ADD/ "Clear and Add"

2. Fixed Point Operations

The following instructions are described; ADD, SUB, ADA, SBA, ADM, SBM, RAA, LWA; ADL, SBL, AAL, SAL; EAD; RND; MPA, MPS; DRJ, ADR, DVJ.

100 ADD IJ A "Add" (0.50 + Ai + Aa)

c(AC)pUA + c(E) → c(AC).

The c(E) (E = #IJA), the addend or the operand, are algebraically added to the UA part of the c(AC). The resulting sum is placed in the AC.

N.B.1. The fixed point operation.

2. The operand is unchanged.

3. Unlike the floating operations, a number in the overflow bit of the operand is neglected in the operation.

4. The whole c(AC), i.e. both the c(UA) and the c(LA) relate to the result.

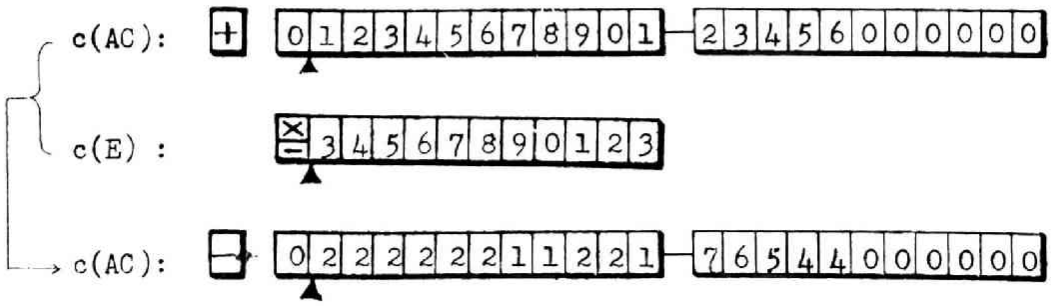
5. A carry or a borrow may occur even to or from a number in the AC overflow position. So an overflow to the AC overflow position may occur. However the result is correct if the result is within the 23 digits of the AC. And if the AC-OV-stop button is set on at this time, the computation stops after the operation, and if set off, the computation does not stop.

6. A real overflow from the AC overflow position may occur. This time the real overflow digit disappears, or the result is a modulo 10 decimal addition. And even if the AC-OV-stop button is set on at this time, the computation no more stops unless a number other than zero is in the AC overflow position.

7. Whenever the result is zero, plus zero(+0) is placed in the AC.

8. In case the "Clear" is attached to the code (the decimal code plus one e.g. 101, ADD/ or "Clear and Add"), the whole c(AC) is cleared or set to +0 before the operation (e.g. ADD). Thus the operation equals to the transfer of the fixed-point form operand to the AC.

e.g. ADD



104 SUB IJ A "Subtruct" (0.50 + Ai + Aa)

c(AC)pUA - c(E) → c(AC).

The c(E) (E = #IJA), the subtrahend or the operand, is algebraically subtracted from the UA part of the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied.

102 ADA IJ A "Add Absolute" (0.50 + Ai + Aa)

c(AC)pUA + |c(E)| → c(AC).

The absolute value of the c(E) (E = #IJA), the addend or the operand, is algebraically added to the UA part of the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied.

106 SBA IJ A "Subtract Absolute" (0.50 + Ai + Aa)

c(AC)pUA - |c(E)| → c(AC).

The absolute value of the c(E) (E = #IJA), the subtrahend or the operand, is algebraically subtracted from the UA part of the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied.

110 ADM .. -- "Add MD" (0.50 + Ai)

c(AC)pUA + C(MD) → C(AC).

The c(MD), the addend or the operand, are algebraically added to the c(AC). The resulting sum is placed in the AC.

N.B.1. The numbers in the index part and the address part of the instruction are neglected in the operation.

114 SBM .. -- "Subtruct MD" (0.50 + Ai)

c(AC)pUA - c(MD) → c(AC).

The c(MD), the subtrahend or the operand, are algebraically subtracted from the UA part of the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied.

2. Any numbers in the index part and the address part of the instruction are neglected in the operation.

130 RAA IJ n "Raise Address" (0.50 + Ai)

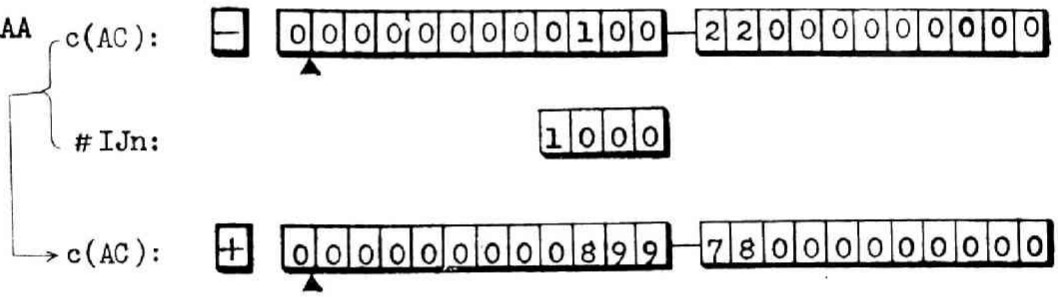
c(AC)pUAad + #IJn → c(AC).

The number #IJn(≡ c(I) + c(J) + n; modulo 10,000), the addend or the operand, is algebraically added to the UA address part of the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied.

2. If a carry occurs as a result of c(I) + c(J) + n, this carry or a fifth digit of the addend is neglected in the operation (the addition of modulo 10,000). So four digits of it are always used.
3. A carry or a borrow (if the angend c(AC) is negative) may occur, e.g. to or from the 5-th position of the UA.

e.g. RAA



134 LWA IJ n "Lower Address" (0.50 + Ai)

c(AC)pUAad - #IJn → c(AC).

A number #IJn (≡ c(I) + c(J) + n; modulo 10,000), the subtrahend or the operand, is algebraically subtracted from the UA address part of the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for RAA(N.B.1 to 3) are applied.

160 ADL IJ A "Add to LA" (0.5 + Ai + Aa)

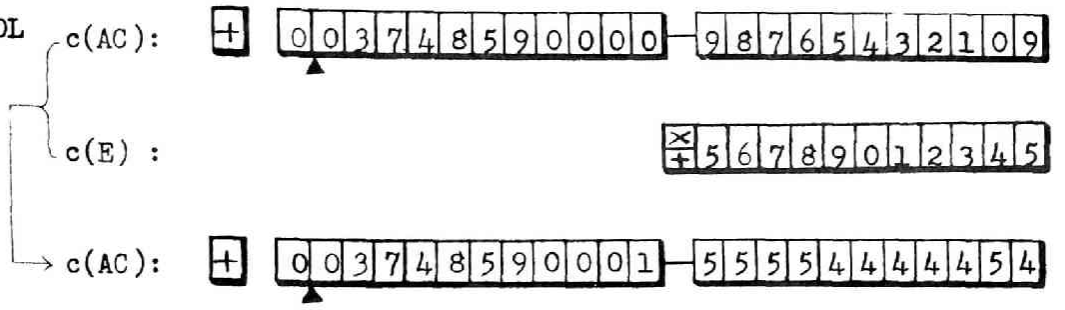
c(AC)pLA + c(E) → c(AC).

The c(E) (E = #IJA), the addend or the operand, are algebraically added to the LA part of the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied.

2. A carry or a borrow may occur, e.g., to or from the least significant digit of c(UA).
3. The LA sign and the LA overflow bit are neglected.

e.g. ADL



164 SBL IJ A "Subtract from LA" (0.55 + Ai + Aa)

c(AC)pLA - c(E) → c(AC).

The c(E) (E = #IJA), the subtrahend or the operand, are algebraically subtracted from the LA part of the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for ADL(N.B.1 and 3) are applied.

162 AAL IJ A "Add Absolute to LA" (0.55 + Ai + Aa)

c(AC)pLA + |c(E)| → c(AC).

The absolute value of the c(E) (E = #IJA), the addend or the operand, is algebraically added to the LA part of the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for ADL(N.B.1 and 3) are applied.

166 SAL IJ A "Subtract Absolute from LA" (0.55 + Ai + Aa)

c(AC)pLA - |c(E)| → c(AC).

The absolute value of the c(E) (E = #IJA), the subtrahend or the operand, is algebraically subtracted from the LA part of the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for ADL(N.B.1 and 3) are applied.

500 EAD IJ A "Extract and Add" (0.55 + Ai + Aa)

c(AC)pUA + c(E) & c(MD)odd → c(AC).

The least significant digit of the c(E) corresponds to that of the

c(MD), the second digit to the second and so on.

A decimal number in each digit of the c(E) (E = #IJA) is regarded to be zero if a decimal number in each corresponding digit of the c(MD) is even, and remains as it is if odd. In other words the c(E) is extracted by the odd numbers in each digit of the MD .

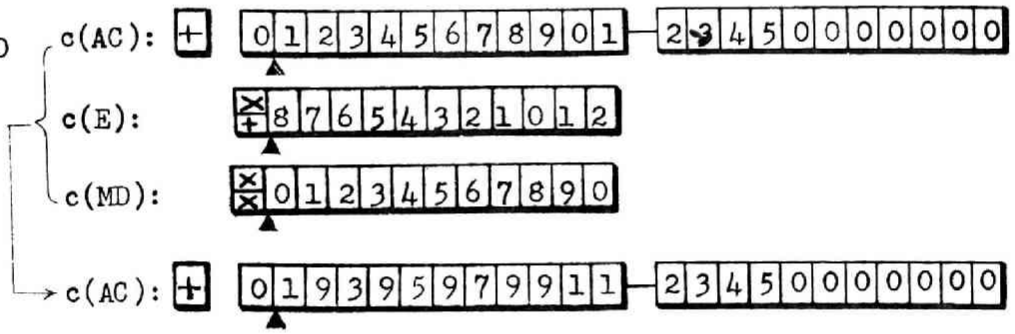
This extracted c(E) as an addend or an operand are algebraically added to the UA part of the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for ADD(N.B.1 to 8) are applied. But "the operand" in the N.B.2 and 3 should be read as "the c(MD) as well as the c(E)" and "the operand" in the N.B.8 should be "the extracted c(E)".

2. The sign of the c(MD) is neglected. The sign of the c(E) is used.

3. An odd number has 1 in the weight 1 of the BCD code, while an even number has 0.

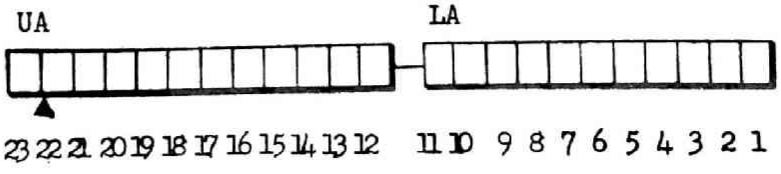
e.g. EAD



138 RND IJ n "Round" (0.50 + Ai)

If the position of the AC which corresponds to the less significant two digits of #IJn (≡ c(I) + c(J) + n; modulo 10,000),(denotedas DU(#IJn), and the correspondency is given below) contains a number greater than 5, the c(AC) are increased by one at the position of the AC which corresponds to DU(#IJn) + 1. If less than 4, unchanged. In either case the less significant digits of the AC than the digit which corresponds to DU(#IJn) (inclusive) are reset to zeros..

The correspondency between the position of the AC and the DU(#IJn) is as follows;



N.B.1. The fixed point operation.

2. The more significant third and the fourth digits of #IJn are neglected.

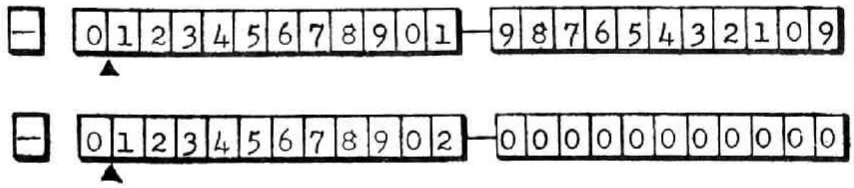
3. If DU(#IJn) is greater than 23, zero replaces the c(AC), but the AC sign is unchanged.

4. If DU(#IJn) = 0, the operation equals to the NOP(514).

5. The AC sign is unchanged in all cases. So c(AC) = -0 may occur.

6. The AC overflow may occur. The N.B.5 and 6 of the ADD(100) are applied except the description of a borrow.

e.g. RND 00 11



120 MPA IJ A "Multiply and Add" (5.8av + Ai + Aa)

c(AC) + c(MD) * c(E) → c(AC).

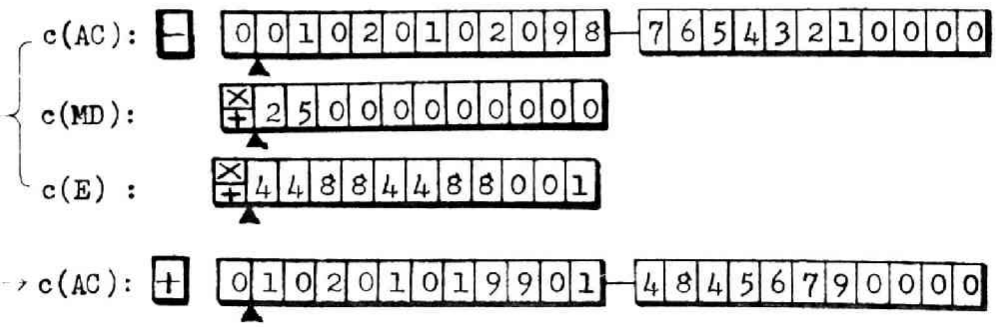
The c(MD) are multiplied by the c(E) (E = #IJA). The product c(MD) * c(E) (22 digits) is algebraically added to the c(AC). The resulting sum is placed in the AC.

N.B.1. The notes of ADD(N.B.1 to 7) are applied. But "the operand" in the N.B.2 and 3 should be read as "the c(MD) as well as the c(E)".

2. In case of "Clear, Multiply and Add, MPA/ or 121", the c(AC) are reset to +0 before the operation MPA. So it becomes; c(MD) * c(E) → c(AC). The product has 22 digits and no AC overflow occurs.

3. The R-indicator, the LA sign and the LA overfolw bit are set off at the beginning of the operation.

e.g. MPA



122 MPS IJ A "Multiply and Subtract" (5.8av + Ai + Aa)

c(AC) - c(MD) * c(E) → c(AC).

The c(MD) are multiplied by the c(E) (E = #IJA). The product

c(MD) * c(E) (22 digits) is algebraically subtracted from the c(AC). The difference replaces the c(AC).

N.B.1. The notes of ADD(N.B.1 to 7) are applied. But "the operand" in the N.B.2 and 3 should be read as "the c(MD) as well as the c(E)".

2. In case of "Clear, Multiply and Subtract, MPS/ or 123," the c(AC) are reset to +0 before the operation MPS. So it becomes; -c(MD) * c(E) →C(AC). The product has 22 digits and no AC overflow occurs.

3. The R-indicator, the LA sign and the LA overflow bit are set off at the beginning of the operation.

152 DRJ IJ A "Divide and Round or Jump" (6.6av + Ai)

c(AC)/c(MD) → c(UA) ; 0 → c(LA), or Jump.

The whole 23-digit c(AC) are treated as a dividend, and the c(MD) as a divisor.

If $|c(AC)| < |c(MD)|$, then division takes place. A 12-digit quotient is calculated. The 12-th or the least significant digit is rounded off, so a 11-digit quotient replaces the c(UA) and zeros replace the c(LA). The AC sign is the algebraic sign of the quotient.

Or if $|c(AC)| \geq |c(MD)|$, then division does not occur but the dividend c(AC) remain unchanged, and the computer takes its next instruction from location E = #IJA, and proceed from there (the control jumps to E).

N.B.1. The fixed point operation.

2. The remainder is not calculated. c.f. DVJ.

3. The c(MD) are unchanged.

4. A number in the overflow bit of the c(MD) is neglected.

5. If there is an overflow in the c(AC) before the operation, the dividend is clearly greater than the divisor, so the control jumps to E. The AC-OV-stop button is ineffective for this case.

6. The AC overflow as a result of the round off of the quotient could occur. If the AC-Ov-stop button is set on at this time, the computation stops after the division and round, and if set off, the computation does not stop.

7. c(UA) = -0 occurs, e.g. if c(AC) = +0 and c(MD) < 0.

8. The R-indicator, the LA sign and the LA overflow bit are set off at the beginning of the operation.

e.g. DRJ c(AC): ⊞ 0 3 4 5 6 7 8 9 0 2 3 3 — 3 3 0 0 0 0 0 0 0 0 0

c(MD): ⊠ 5 0 0 0 0 0 0 0 0 0 0

→ c(AC): ⊞ 0 6 9 1 3 5 7 8 0 4 6 7 — 0 0 0 0 0 0 0 0 0 0 0

140 ADR IJ A "Add, Divide and Round or Halt" (6.9av + Ai + Aa)

c(AC)pUA + c(E) /c(MD) → c(UA); 0 — c(LA), or Halt.

The c(E) (E = #IJA) are algebraically added to the UA part of the c(AC). The resulting sum is placed in the AC. This sum or the whole c(AC) is treated as a dividend, and the c(MD) as a divisor.

If $|c(AC)pUA + c(E)| < |c(MD)|$, then division takes place. A 12-digit quotient is calculated. The 12-th or the least significant digit is rounded off, so a 11-digit quotient replaces the c(UA) and zeros replace the c(LA). The AC sign is the algebraic sign of the quotient.

Or if $|c(AC)pUA + c(E)| \geq |c(MD)|$, division does not occur and the computation stops with the HALT lamp on. The dividend remains unchanged in the AC.

N.B.1. The operation is a composite operation of ADD(100), and DRJ(152) except the difference of jump (DRJ) from halt (ADR).

2. The notes for the first part of the operation (i.e. ADD) is therefore the same as that for ADD(N.B.1 to 8) except the description concerning with the AC overflow, or a dividend is clearly greater than the divisor, the division does not take place and the computation stops.

3. The notes for the second part of the operation (i.e. DRJ) is therefore the same as that for DRJ(N.B.1 to 8).

4. In case of "Clear, Add, Divide and Round or Halt", the c(AC) are reset to +0 before the operation ADR. So it becomes;
c(E)/c(MD) → c(UA); 0 → c(LA), or Halt.

150 DVJ IJ A "Divide or Jump" (6.0av + Ai)

c(AC)/c(MD) → c(UA); R → c(LA), or Jump.

The R-indicator, the LA sign and the LA overflow bit are set off at the

beginning of the operation.

The whole c(AC) are treated as a dividend, and the c(MD) as a divisor.

If $|c(AC)| < |c(MD)|$, then division takes place. A 11-digit quotient is calculated and it replaces the c(UA). And its remainder, i.e. a 11-digit fraction $\{c(AC) - c(MD) * (\text{quotient})\} * 10^{11}$ whether zero or not replaces the c(LA). The AC sign is the algebraic sign of the quotient. The sign of the dividend replaces the LA sign as of remainder and the R-indicator is set on again.

Or if $|c(AC)| \geqq |c(MD)|$, then division does not occur but the dividend c(AC) remain unchanged, and the computer takes its next instruction from location E (=#IJA) and proceed from there (the comtrol Jumps to E). The R-indicator is not set on this time.

N.B.1. The fixed point operation.

2. The c(MD) are unchanged.

3. A number in the overflow bit of the c(MD) is neglected in the operation.

4. As the result of the division no AC overflow occurs.

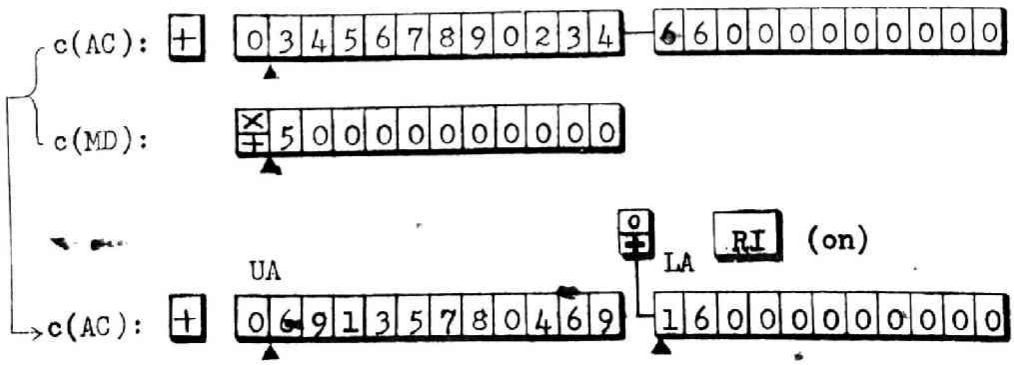
5. If the AC overflow exists before the operation, it is clear $|c(AC)| \geqq |c(MD)|$, and the control jumps to E. The AC-OV-stop button is ineffective.

6. c(UA) = -0 occures, e.g. if c(AC) = +0 and c(MD) < 0.

7. The remainder in the LA is stored only by the SLA (302). (c.f. SLA) Since e.g. by shifting instructions the LA sign can not be transferred.

8. The R-indicator, the LA sign and the LA overflow bit are set off at the beginning, and set on only when division is made, and keep on until any next multiplication, division, or the "Clear" c(AC) operation.

e.g. DVJ



3. Floating Point Operations

The following instructions are described; FAD, FSB, FAA, FSA, FAM, FSM; FRD; FMP/, FMC/; FDR, FAV, FDJ.

The arithmetic operations of floating numbers are analogous to the ordinary arithmetic operations with the decimal points. All numbers are however expressed in the floating-point form.

All the results obtained by the floating-point arithmetic operations are ordinarily in the normalized form of floating-point numbers except a remainder obtained by FDJ.

200 FAD IJ A "Floating Add" (1.3 + Ai + Aa)

c(AC) + c(E) → c(AC).

The c(E) (E = #IJA), the addend or the operand, are algebraically added to the c(AC). The reaulting sum is placed in the AC.

N.B.1. The floating point operations.

2. The operand is unchanged.

3. Unlike the fixed point operations, a number in the overflow bit of the operand is used as a third digit (one hundred) of a characteristic.

4. The whole c(AC), i.e. both c(UA) and c(LA), relate to the result. But numbers in the 1' and 2' positions of LA are set off to zero throughout the operation. i.e. The mantissa of the c(AC) is 18 digits or of the double precision.

5. The execution of the operation is briefly as follows; If the mantissa of either the operand or the c(AC) is zero, a number with non-zero mantissa replaces the c(AC) and the normalization takes place. If both zeros, the c(AC) is cleared. Or if not the above, then the characteristic of the c(AC) is compared with that of the operand. The mantissa of a number with less characteristic is shifted to the right (numbers which underflow from the 3' position of the LA or its equivalent disappear) with a compensating change in the characteristic and corresponding zeros are supplied in the vacated positions until the both characteristics become the same. The ordinary algebraic addition (or subtraction for subtract orders) is then performed within the length of the 18 digits. Then the normalization is made, i.e. the mantissa of the result becomes a fraction without the heading zeros and within the length of 18 digits.

6. The exponent overflow or underflow (modulo 1,000 addition or subtraction) may occur. However, the result is correct. And if the AC-EX-OV-stop button is set on at this time, the computation stops

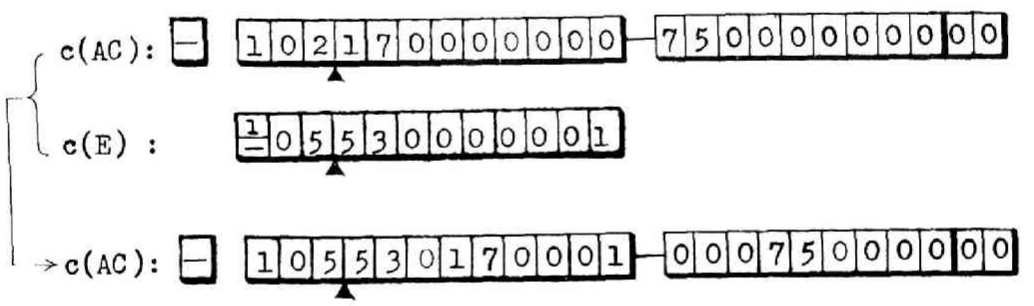
after the operation, and if set off, the computation does not stop.

7. If exponent overflow of the c(AC) has been occured before the operation, wrong answer is calculated and the computer treats it as a result of the operation.

8. Whenever the result is zero, plus zero (+0) is placed in the AC, which is the same form as that of a fixed point number.

9. In case the "Clear" is attached to the code (the decimal code plus 1, e.g. 201, FAD/ or "Clear and Floating Add"), the whole c(AC) is cleared or set to +0 before the operation (e.g. FAD). Thus the operation equals to the transfer of the floating-point form operand to the AC and the normalization of it.

e.g. FAD



204 FSB IJ A "Floating Subtract" (1.3 + Ai + Aa)

c(AC) - c(E) → c(AC).

The c(E) (E=#IJA), the subtrahend or the operand, are algebraically subtracted from the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for FAD(N.B.1 to 9) are applied.

202 FAA IJ A "Floating Add Absolute" (1.3 + Ai + Aa)

c(AC) + |c(E)| → c(AC).

The absolute value of the c(E) (E - IJA), the addend or the operand, is algebraically added to the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for FAD(N.B.1 to 9) are applied.

206 FSA IJ A "Floating Subtract Absolute" (1.3 + Ai + Aa)

c(AC) - |c(E)| → C(AC).

The absolute value of the c(E) (E - IJA), the subtrahend or the operand, is algebraically subtracted from the c(AC). The difference replaces the c(AC).

N.B.1. The same notes as for FAD(N.B.1 to 9) are applied.

210 FAM .. - "Floating Add MD" (1.3 + Ai)

c(AC) + c(MD) → c(AC).

The c(MD), the addend or the operand, are algebraically added to the c(AC). The resulting sum is placed in the AC.

N.B.1. The same notes as for FAD(N.B.1 to 9) are applied.

2. Any numbers in the index part and the address part of the instruction are neglected in the operation.

214 FSM .. - "Floating Subtract MD" (1.3 + Ai)

c(AC) - c(MD) → c(AC).

The c(MD), the subtrahend or the operand, are algebraically subtracted from the c(AC). The difference replaces the c(AC).

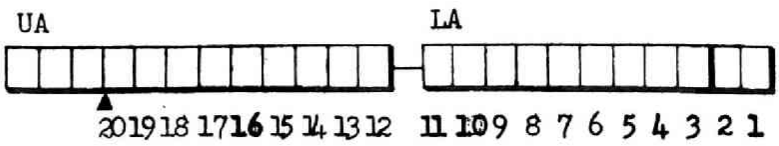
N.B.1. The same notes as for FAD(N.B.1 to 9) are applied.

2. Any numbers in the index part and the address part of the instruction are neglected in the operation.

238 FRD IJ n "Floating Round" (0.55 + Ai)

If the position of the AC, which correspond to the less significant two digits of IJn (≡ c(I) + c(J) + n; modulo 10,000), (denoted as DU(#IJn), and the correspondency is given below) contains a number greater than 5, the c(AC) are increased by one at the position of the AC which corresponds to DU(#IJn) + 1. If less than4, unchanged. In either case the less significant digits of the AC than the digit which corresponds to DU(#IJn) (inclusive) are reset to all zero. And if a carry occurs from the most significant digit of the mantissa as a result of the round off, the mantissa is shifted to the right by one with the corresponding compensation of the characteristic (i.e. one is added to it).

The correspondency between the position of the AC and the DU(#IJn) is as follows;



N.B.1. **The floating point operation.**

2. The more significant third and the fourth digits of #IJn are neglected.

3. If DU(#IJn) $\geq$ 21, the operation equals to the RND (138).

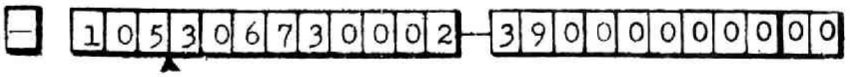
4. If DU(#IJn) = 0, the operation equals to the NOP (514).

5. The AC sign is unchanged in all cases.

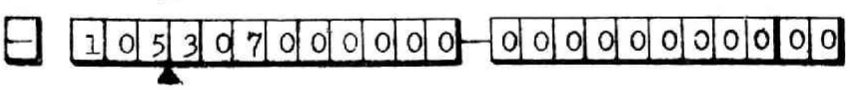
6. The exponent overflow may occur. The N.B. 6 and 7 of the FAD (200) are applied.

7. The normalization is not made except the way described above.

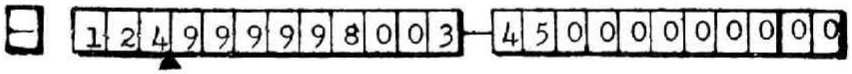
e.g. FRD 00 17



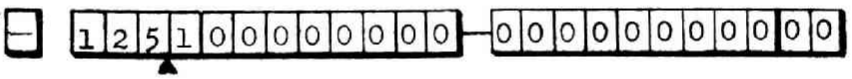
c(AC):



FRD 00 15



c(AC):



221 FMP/ IJ A "Clear and Floating Multiply" (5.2av + Ai + Aa)

c(MD) * c(E) → c(AC).

The c(MD) are multiplied by the c(E) (E = #IJA). The product c(MD) * c(E) is placed in the AC.

N.B.1. The floating point operation.

2. The both c(MD) and c(E) are unchanged.

3. Unlike the fixed point operations, a number in the overflow bit of the c(MD) as well as of the c(E) is used as a third digit (one hundred) of a characteristic (or an exponent +100).

4. At the end of the operation the normalization is made, so the product is in normalized form, i.e. the mantissa of the result becomes a fraction without the heading zeros and within the length of 18 digits.

5. The exponent overflow or underflow (modulo 1,000 addition and subtraction respectively at the characteristic part of the AC) may occur. The N.B.6 and 7 of the FAD (200) are applied.

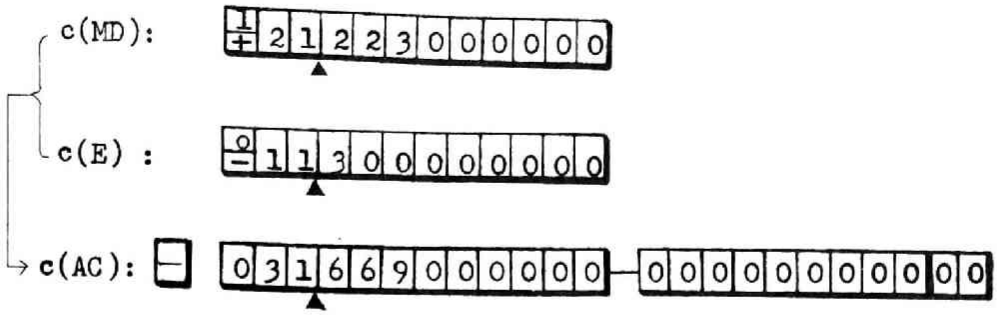
6. The AC sign is the algebraic sign of the product, so c(AC) = -0 may occur.

7. FMP alone does not exist.

8. The R-indicator, the LA sign and the LA overflow bit are set off at

the beginning of the operation.

e.g. FMP/



223 FMC/ IJ A "Floating Multiply and Change Sign" (5.2av + Ai + Aa)

$-c(MD) * c(E) \rightarrow c(AC)$.

The c(MD) are multiplied by the c(E) (=#IJA). Then the sign of the product is changed and the result is placed in the AC.

N.B.1. The operation is a composite operation of the FMP/ (221) and CHS (512). For the first part of the operation i.e. FMP/, the same notes as for FMP/ are applied.

252 FDR IJ A "Floating Divide and Round or Jump" (5.6av + Ai)

$c(AC) / c(MD) \rightarrow c(UA)$; $0 \rightarrow c(LA)$, or Jump.

The c(AC) are treated as a dividend, and the c(MD) as a divisor.

If the c(MD) are zero or not in a normalized form (i.e. the mantissa has at least a heading zero), division does not occur, but the dividend c(AC) remain unchanged, and the computer takes its next instruction from the location E (=#IJA) and proceed from there (the control jumps to E).

Or if it is not the above case, then division takes place. A 10-digit quotient is calculated. The 10-th or the least significant digit is rounded off, so a quotient of 9-digit mantissa and 3-digit characteristic replaces the c(UA). Zero replaces the c(LA). The AC sign is the algebraic sign of the quotient.

N.B.1. The floating point operation.

2. The remainder is not calculated, c.f. FDJ.

3. The c(MD) and the c(E) are unchanged.

4. Unlike the fixed point operations, a number in the overflow bit of the c(MD) as well as of the c(E) is used as a third digit (one hundred)

of a characteristic (or an exponent +100).

5. Numbers in the 1' and 2' positions of the LA are set off to zero at the beginning of the operation.

6. The exponent overflow or underflow (modulo 1,000 addition or subtraction) may occur as a result of the operation. The result is however correct. If the AC-EX-OV-stop button is set on at this time, the computation stops after the operation, and if set off, the computation does not stop.

7. If exponent overflow of the c(AC) has been occured before the operation, wrong answer is calculated and the computer treats it as a result of the operation.

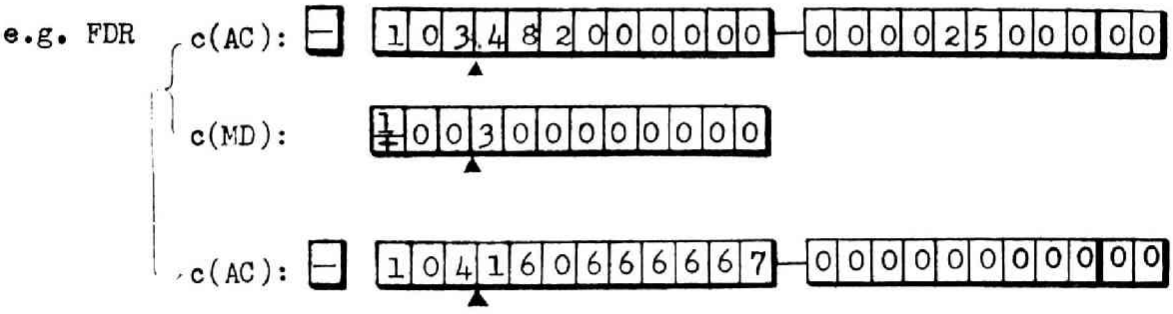
8. The quotient is given in the normalized form, if the dividend (as well as the divisor) has been given in a normalized form. Unless the above, the quotient is not always given in the normalized form.

9. The execution of the division is briefly as follows; the mantissa of the c(MD) (denoted as md) is checked if it has a heading zero, if so, the control jumps to E. Next, the md is compared with that of the c(AC) (denoted as ma). If ma $<$ md, division takes place. If ma $\geq$ md, the ma is shifted to the right by one and one is added to its characteristic, and division takes place. Next round off is made, and if a carry occurs from the most significant digit of the quotient, the result is shifted to the right by one and one is added to its characteristic.
The calculations of the characteristics are all in the modulo 1,000 fashion (c.f. N.B.7).
Then the form of the quotient is what is described in the above N.B.8.

10. c(UA) = -0 occurs, e.g. if the dividend is +0 and c(MD) $<$ 0.

11. The R-indicator, the LA sign and the LA overflow bit are set off at the beginning of the operation.

e.g. FDR



240 FAV IJ A "Floating Add, Divide and Round or Halt" (6.7av + Ai + Aa)

$\{c(AC) + c(E)\} / c(MD) \rightarrow c(UA)$; $0 \rightarrow c(LA)$, or Halt

The c(E) (E = #IJA) are algebraically added to the c(AC). The resulting sum is placed in the AC. This sum or the whole c(AC) is treated as a dividend, and the c(MD) as a divisor.

If the c(MD) are zero or not a normalized form (i.e. the mantissa has a heading zero.), division does not occur and the computation stops with the HALT lamp on. The dividend c(AC) + c(E) remains unchanged in the AC.

Or if it is not the above case, then division takes place. A 10-digit quotient is calculated. The 10-th or the least significant digit is rounded off, so a quotient of 9-digit mantissa and 3-digit characteristic replaces the c(UA), and zero replaces the c(LA). The AC sign is the algebraic sign of the quotient.

N.B.1. The operation is a composite operation of FAD (200), and FDR (252) except the difference of jump (FDR) from halt (FAV).

2. The notes for the first part of the operation (i.g. FAD) is therefore the same as that for FAD (N.B.1 to 9) except the description of the AC-EX-OV-stop button in the N.B.6. Even if the overflow or underflow occurs as a result of the addition (FAD) of FAV, division process (FDR) follows and the overall result of FAV is still correct.

3. The notes for the second part of the operation (i.e. FDR) is therefore the same as that for FDR(N.B.1 to 11). But in case of FAV, the dividend is always in the normalized form, so is the quotient (N.B.8).

4. In case of "Clear, Floating Add and Divide or Halt", the c(AC) are set to +0 before the operation FAV. So it becomes;
c(E) / c(MD) → c(UA); 0 → c(LA), or Halt.

250 FDJ IJ A "Floating Divide or Jump" (5.law + Ai)

c(AC) / c(MD) → c(UA); R → c(LA), or Jump.

The R-indicator, the LA sign and the LA overflow bit are set off at the beginning of the operation. The whole c(AC) are treated as a dividend, and the c(MD) as a divisor.

If the c(MD) are zero or not in a normalized form (i.e. the mantissa has at least a heading zero), division does not occur, but the dividend remains unchanged, and the computer takes its next instruction from the location E (= #IJA) and proceed from there (Jump E).

Or if it is not the above case, then division takes place. A quotient with 9-digit mantissa and 3-digit characteristic is calculated and replaces the c(UA), And its remainder with 9-digit mantissa and 3- digit (precisely

the LA overflow bit and m' and 10' position of the LA) characteristic, whether zero or not, i.e. c(AC) - c(MD) * (quotient), replace the c(LA). The AC sign is the algebraic sign of the quotient. The sign of the dividend replaces the LA sign as of the remainder, and the R-indicator is set on again.

N.B.1. The floating point operation.

2. The remainder is calculated, and the quotient is not rounded off.

3. The c(MD) are unchanged.

4. Unlike the fixed point operations,a number in the overflow bit of the c(MD) is used as a third digit (one hundred) of a characteristic (or an exponent +100).

5. Numbers in the 1' and 2' positions of the LA are set off to zero at the beginning of the operation.

6. The exponent overflow or underflow (modulo 1,000 addition or subtraction) of the quotient may occur. The quotient is however correct. The exponent underflow of the remainder (modulo 200 subtraction) may occur. At this time the characteristic of the remainder is no more correct. (c.f. N.B.10).
If the AC-EX-OV-stop button is set on at either or both of the above two cases (i.e. that of the quotient or of the remainder), the computation stops after the operation, and if set off, the computaion does not stop.

7. If exponent overflow or underflow of the c(AC) has been occured before the operation, wrong answer is calculated and the computer treats it as a result of the operation.

8. The quotient is given in the normalized form, if the dividend (as well as the divisor) has been given in a normalized form. Unless the above, the quotient is not generally given in the normalized form.

9. The remainder is not generally given in the normalized form. e.g. The mantissa is zero and the characteristic is not zero often occurs.

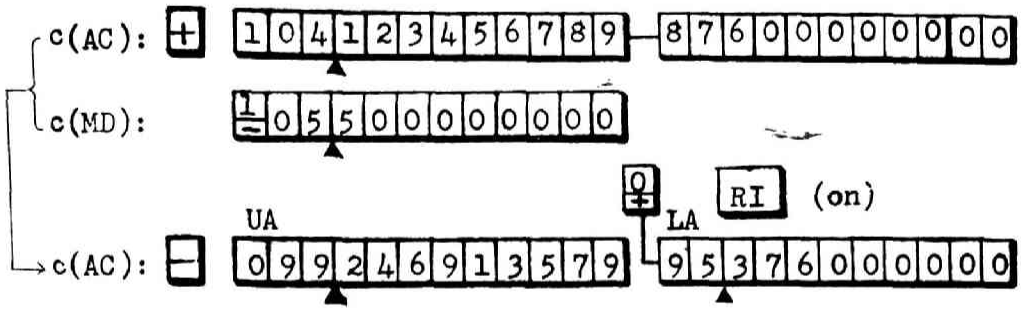
10. The execution of the division is briefly as follows; The mantissa of the c(MD) (denoted as md) is compared with that of the c(AC) (denoted as ma). If ma < md, division takes place. If ma $\geq$ md, the ma is shifted to the right by one and one is added to its characteristic, and division takes place. Then 9-digt quotient is calculated.
The value of the characteristic of the c(MD) is subtracted from that of the c(AC) (or as in the above, that increased by 1) and 100 is added to it. The result replaces the characteristic of the c(AC) as that of the quotient. All calculations are in modulo 1,000 fashion.
For the characteristic of the remainder, 9 is subtracted from the characteristic of the c(AC) (or as in the above that increased by 1) in modulo 200, and this difference is used (N.B.6).
Then the form of the quotient as well as of the remainder becomes what is described in the above N.B. 8 and 9.

11. c(UA) = -0 occurs, e.g. if the dividend is +0 and c(MD) < 0.

12. The remainder in the LA is stored only by the SLA (302) (c.f. SLA) not by the FSL.

13. The R-indicator, the LA sign and the LA overflow bit are set off at the beginning, and set on only when division is made, and keeps on until any next multiplication or division instructions, or the "Clear" c(AC) operation.

e.g. FDJ



4. Conversion Operations

The following instructions are described; FFL, FFX.

410 FFL .. - "Fixed to Floating" (0.80 + Ai)

A fixed point number in the 23-digit AC is converted into a normalized floating point number and the result is set in the AC.

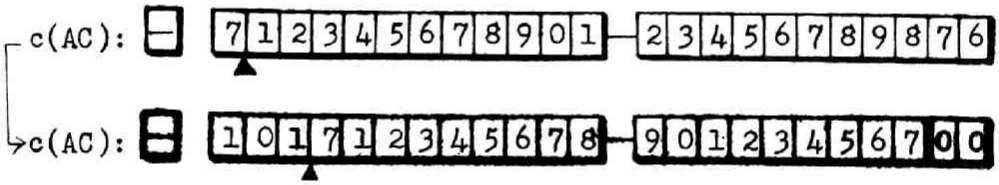
N.B.1. The AC sign is unchanged.

2. The overflow digit of the UA (integer) is also included in the operation.

3. At the end of the normalization (i.e. at the end of the operation) the 1' and 2' positions of the LA are set off to zero. So a 23-digit number is reduced to a 18-digit mantissa.

4. Any numbers in the index part and the address part of the instruction are neglected in the operation.

e.g. FFL



450 FFX IJ A "Floating to Fixed or Jump" (0.55 + Ai)

A floating point number in the AC is converted into a fixed point number and the result is placed in the AC, if the value of the characteristic of the c(AC) is in ~~the~~ range between 100 and 0. Or if in the range between 101 and 199 (the c(AC) is equal to or greater than one in the normalized form), the c(AC) are unchanged and the computer takes its next instruction from the location E (= #IJA) and proceed from there (the control jumps to E).

N.B.1. The AC sign is unchanged.

2. The less significant digits than the 1' position of the LA in its fixed point form disappears. So e.g. c(AC) = -0 as a result may occur.

3. The AC overflow does not occur.

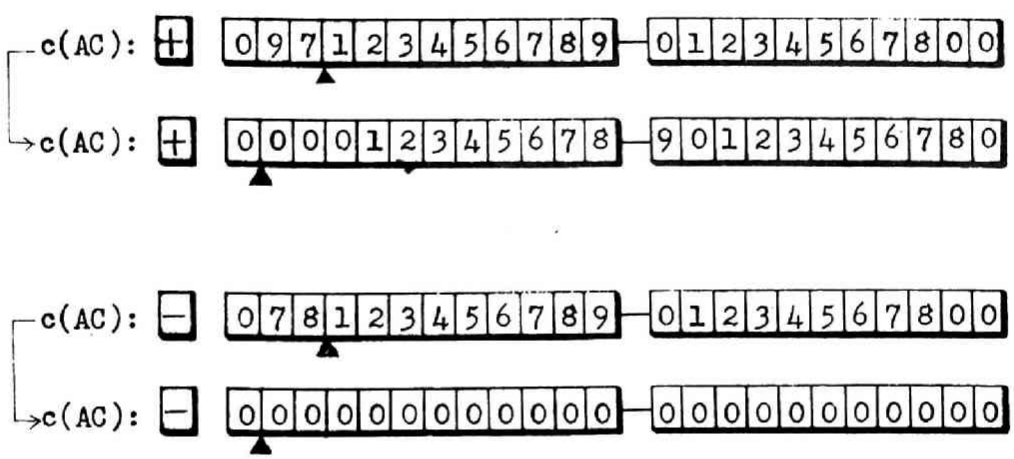
4. If the AC exponent overflow or underflow has been occured before the operation, wrong answer is calculated and the computer treats it as a result of the operation.

5. If the c(AC) has not been in the normalized form before the operation,

the result is still correct. (c.f. N.B.6)

6. The execution of the operation is briefly as follows; If the mantissa of the c(AC) is zero, the c(AC)are cleared. Or if not the above, the characteristic of the c(AC) is examined. If it is greater than 100 and smaller than 200 (i.e. 101-199), "jump to E" is made. Or if not the above, the mantissa is shifted to the right and zeros are supplied to the vacated positions until the its characteristic is compensated and reaches 100. And 22 digits of the above replaces the c(AC).
In case of the AC exponent overflow or underflow, the way of miscalculation is rather in case by case fashion, so it is not described here.

e.g. FFX



5. Sign Part Operations

The following instructions are described; SSP, CHS.

510 SSP .. - "Set Sign Plus" (0.30 + Ai)

$|c(AC)| \rightarrow c(AC)$

The AC sign is set plus.

N.B.1. The absolute value of the c(AC) replaces the c(AC).

2. Any numbers in the index and the address part of the instruction are neglected in the operation.

3. The LA sign is unchanged.

512 CHS .. - "Change Sign" (0.30 + Ai)

$-c(AC) \rightarrow c(AC)$

The AC sign is changed, i.e. + is changed to -, - is to +.

N.B.1. The magnetude of the c(AC) is unchanged.

2. If CHS/, -0 replaces the c(AC).

3. Any numbers in the index and the address part of the instruction are neglected in the operation.

4. The LA sign is unchanged.

6. Shifting Operations

The following instructions are described; SLS, LLS, SRS, LRS, LCS.

The instruction SCT is explained in the Special Operations.

The number of places of shift is specified by the less significant two digits of #IJn ($\equiv$ c(I) + c(J) + n; module 10,000), denoted as DU(#IJn). The more significant third and the fourth digits of #IJn are neglected.

Numbers which overflow the capacity of a register disappear, and zeros are supplied in the vacated positions of a register with the exception of LCS.

The AC sign is unchanged. Thus -0 may occur.

530 SLS IJ n "Short Left Shift" (0.55 + Ai)

The 12-digit c(UA) are shifted to the left by the number of places specified by the DU(#IJn).

N.B.1. The c(LA) are unchanged.

2. If DU(#IJn) $\geq$ 12, the c(UA) are replaced by zeros.

e.g.

SLS 00 3

c(AC): − 1 2 3 4 5 6 7 8 9 0 1 2 | 3 4 5 6 7 8 9 0 1 2 3

c(AC): − 4 5 6 7 8 9 0 1 2 0 0 0 | 3 4 5 6 7 8 9 0 1 2 3

532 LLS IJ n "Long Left Shift" (0.55 + Ai)

The 23-digit c(AC) are shifted to the left by the number of places specified by the DU(#IJn).

N.B.1. If DU(#IJn) $\geq$ 23, the c(AC) are replaced by zeros.

e.g.

LLS 00 3

c(AC): + 1 2 3 4 5 6 7 8 9 0 1 2 | 3 4 5 6 7 8 9 0 1 2 3

c(AC): + 4 5 6 7 8 9 0 1 2 3 4 5 | 6 7 8 9 0 1 2 3 0 0 0

536 SRS IJ n "Short Right Shift" (0.55 + Ai)

The 12-digit c(UA) are shifted to the right by the number of places

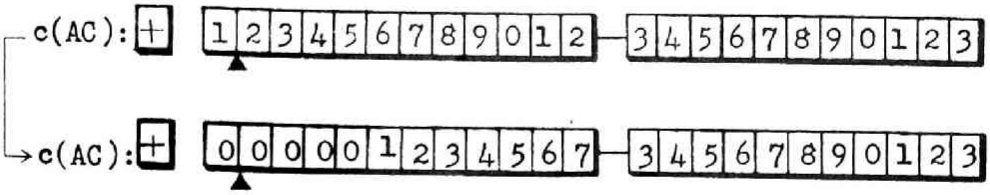
specified by the DU(#IJn).

N.B.1. The c(LA) are unchanged. Numbers which underflow from the 1 position of the UA disappear.

2. If DU(#IJn) $\geq$ 12, the c(UA) are replaced by zeros.

e.g.

SRS 00 5



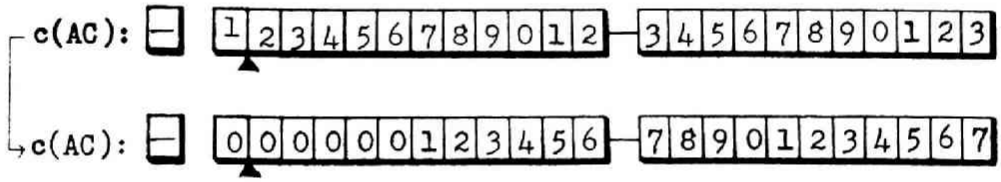
538 LRS IJ n "Long Right Shift" (0.55 + Ai)

The 23-digit c(AC) are shifted to the right by the number of places specified by the DU(#IJn).

N.B.1. If DU(#IJn) $\geq$ 23, the c(AC) are replaced by zeros.

e.g.

LRS 00 6



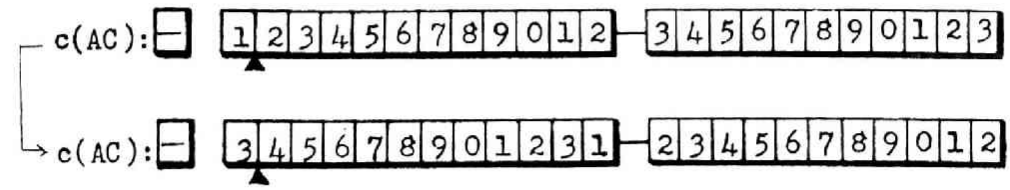
534 LCS IJ n "Long Cyclic Shift" (0.55 + Ai)

The 23-digit c(AC) are shifted to the left by the number of places specified by the DU(#IJn). But a number in the AC overflow position is shifted into the 1' position of the LA. Thus the shift becomes a cyclic shift.

N.B.1. If DU(#IJn) $\geq$ 23, the c(AC) are replaced by zeros.

e.g.

LCS 00 12



7. Word Transfer Operations

The following instructions are described; STO, STM, LDM, PAM; SLA, FSL.

300 STO IJ A "Store" (0.35 + Ai + Aa)

c(UA) → c(E).

Numbers in positions from 1 to 11 of the UA and in the weight 1 (BCD code) of the AC overflow position and the AC sign replace the c(E) (E = #IJA). In other words the c(UA) except numbers in the weight 2, 4 and 8 of the AC position replace the c(E).

N.B.1. The instruction can be used for both fixed and floating numbers.

2. Any odd number in the AC overflow position set the overflow bit of the c(E) with 1.

3. The c(UA) are unchanged.

4. 301 STO/ IJ A "Clear and Store"
The both c(AC) and the c(E) are reset to +0.

304 STM IJ A "Store MD" (0.35 + Ai + Aa)

c(MD) → c(E).

The c(MD) replace the c(E).

N.B.1. This instruction can be used for both fixed and floating numbers.

2. A number zero or one in the overflow bit of the MD also replaces that of the location E.

3. The c(MD) are unchanged.

320 LDM IJ A "Load MD" (0.35 + Ai + Aa)

c(E) → c(MD).

The c(E) replace the c(MD).

N.B.1. This instruction can be used for both fixed and floating numbers.

2. A number zero or one in the overflow bit of the location E also replaces that of the MD.

3. The c(E) are unchanged.

310 PAM .. - "Place UA to MD" (0.35 + Ai) ...

c(UA) → c(MD).

Numbers in positions from 1 to 11 of the UA and in the weight 1 (BCD code) of the AC overflow position and the AC sign replace the c(MD). In other words the c(UA) except numbers in the weight 2, 4 and 8 of the AC overflow position replaces the c(MD).

N.B.1. The instruction can be used for both fixed and floating numbers.

2. Any odd number in the AC overflow position set the overflow bit of the c(MD) with 1.

3. The c(UA) are unchanged.

4. Any numbers in the index part and the address part of the instruction are neglected in the operation.

302 SLA IJ A "Store LA" (0.55 + Ai + Aa)

c(LA) → c(E) [R]

If the R-indicator is off, the c(LA) with the AC sign replace the c(E) (E = #IJA), * Or if on, the c(LA) with the LA sign and the LA overflow bit replace the c(E). * and an overflow bit of the c(E) is reset to zero.

N.B.1. The R-indicator is on only when division is made by either DVJ or FDJ, i.e. the remainder replaces the c(LA).

2. The fixed or floating-point form remainder by either DVJ or FDJ can be stored only by this instruction and only when R-indicator is on. This instruction does not set off the R-indicator (c.f. DVJ and FDJ).

3. The R-indicator is set off only when "Clear" operation of the c(AC), multiplication or division instructions are executed.

4. So this instruction can be used for fixed point number if the R-indicator is off (at this time FSL is used for floating point numbers), and if on, for both fixed and floating point remainders.

5. The c(AC) as well as the LA sign and the LA overflow bit and the R-indicator is unchanged.

340 FSL IJ A "Floating Store LA" (0.95av + Ai + Aa)

c(LA) → c(E).

Numbers in the positions from m' to 3' of the LA as a 9-digit mantissa, the value of the characteristic of the AC minus 9 as a 3-digit characteristic

and the AC sign form a floating point number. Then it is normalized and the result replaces the c(E).

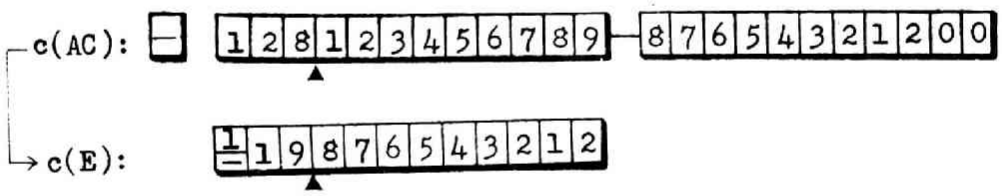
N.B.1. The instruction is used only for floating numbers.

2. The c(AC) are unchanged.

3. The remainder as a result of FDJ can not be stored by this instruction (c.f. SLA). So the R-indicator, the LA sign and the LA overflow bit are neglected.

4. The exponent underflow (modulo 200 subtraction) may occur. The result is not correct. If the AC-EX-OV-stop button is set on at this time, the computation stops. Or if off, does not stop.

e.g. FSL



8. Address Transfer Operations

The following instructions are described; STA, LDA, STL.
Other two instructions RAA and LWA are explained in the Fixed Point Arithmetic Operations.

306 STA IJ A "Store Address" (0.35 + Ai + Aa)

c(UA)ad → c(E)ad.

The address part of the c(UA) replaces that part of the c(E) (E = #IJA).

N.B.1. The c(E) except that of the address part, and the c(AC) are unchanged.

2. If the "Clear" c(AC) is attached; +0 → c(AC); 0 → c(E)ad.

307 LDA IJ A "Load Address" (0.35 + Ai + Aa)

c(E)ad → c(UA)ad.

The address part of the c(E) (E= #IJA) replaces that part of the (UA).

N.B.1. The c(AC) except that of the address part and the c(E) are unchanged.

2. If the "Clear" c(AC) is attached; c(E)ad → c(UA)ad and the other parts of the c(AC) become zeros and the AC sign becomes plus.

308 STL IJ A "Store Location" (0.35 + Ai + Aa)

c(LC) → c(E) ad.

The c(LC), i.e. the address number where this instruction is stored, replaces the address part of the c(E) (E = #IJA).

N.B.1. The c(E) except the address part are unchanged.

9. Block Transfer Operations

The following instructions are described; LDQ, STQ; BDM, DMB.

The block transfer instructions transfer a group of words which consists of fifty (ordinarily) or less than fifty words between portions of the memory whose access time is different, i.e. between the normal and the quick access memory of the drum (STQ and LDQ), or between the drum memory and the core memory (BDM and DMB), or between the core memory and the magnetic tape memory (this part is described in the magnetic tape operations).

The quick access memory of the drum consists of 4 bands, each of which has the same access time (av ca. 1.25ms) and is numbered from 1 to 4. The both instructions LDQ and STQ specify the number of the bands by the Xx position of the instruction, and at this time it will be denoted as Q. The correspondency between numbers of the bands and the memory address is as follows;

Q = 1	locations from 4,000 to	4,049
2	4,050	4,099
3	4,100	4,149
4	4,150	4,199

364 LDQ QJ A "Load Quick Access" (2.90 + Ai + Aa)

c(E), ..., c(EL) → c(QE), ..., c(QEL).

E = #JA, $0 \leqq EL - E < 50$, $EL \equiv QEL \equiv 49$ in modulo 50,

$0 \leqq QEL - QE < 50$, and QE in No.Q band.

The c(E) replace the c(QE), ..., and the c(EL) replace the c(QEL). Thus a group of words is transferred to the quick access memory of No. Q band from the normal access memory.

N.B.1. Contents of any other register than the c(QE), ..., c(QEL) are unchanged. So c(QE - 1) etc. are unchanged.

2. If $5 \leqq Q \leqq 9$ or Q = 0, the operation is equal to NOP (514).

3. If $4000 \leqq E \leqq 9999$, the c(E), ..., c(EL) are regarded as all zeros.

366 STQ QJ A "Store Quick Access" (2.90 + Ai + Aa)

c(QE), ..., c(QEL) — c(E), ..., c(EL).

For E, EL, Q and QEL, the instruction LDQ should be referred to. The c(QE) replace the c(E), ..., and the c(QEL) replace the c(EL). Thus, a group of words is transferred to the normal access memory from the quick access memory of No. Q band.

N.B.1. Contents of any other register than the c(E),..., c(EL) are unchanged.

2. If $5 \leqq Q \leqq 9$ or $Q = 0$, the operation is equal to NOP (514).

3. If $4000 \leqq E \leqq 9999$, the operation is equal to NOP (514).

920 DMB ,J A "Drum to Buffer" (2.90 + Ai + Aa)

c(E), ..., c(EL) → c(4200), ..., c(4200 + EL - E).

$E = \#JA$, $0 \leqq EL - E < 50$, and $EL \equiv 49$ in modulo 50.

The c(E) replace the c(4200), ..., and the c(EL) replace the c(4200 + EL - E). Thus a group of words is transferred to the core memory from the drum memory (both normal and quick access).

N.B.1. Contents of any other registers than the c(4200), ..., c(4200 + EL - E) are unchanged.

2. If $4200 \leqq E \leqq 9999$, zeros replace the c(4200), ..., c(4200 + EL - E).

3. A number in the Xx part is neglected in the operation.

4. If the core memory has been busy by some tape operations, the execution is done after the busy is off.

922 BDM ,J A "Buffer to Drum" (2.90 + Ai + Aa)

c(4200), ..., c(4200 + EL - E) → c(E), ..., c(EL).

$E = \#JA$, $0 \leqq EL - E < 50$, and $EL \equiv 49$ in modulo 50.

The c(4200) replace the c(E), ..., and the c(4200 + EL - E) replace the c(EL). Thus a group of words is transferred to the drum memory (both normal and quick access) from the core memory.

N.B.1. Contents of any other registers than the c(E), ..., c(EL) are unchanged.

2. If $4200 \leqq E \leqq 9999$, the operation equals to NOP (514).

3. A number in the Xx part is neglected in the operation.

4. If the core memory has been busy by some tape operations, the execution is done after the busy is off.

10. Control Operations

The following instructions are described; NOP; HJM, JMP, JMI, JUN, JNZ, JOV, JEO, JSW; JXL, JXR, JXU, JSX; CMP.

Other two control instructions JTG and JTE are explained in the Magnetic Tape Operations.

The control instruction causes an alteration of a normal sequence of a program, i.e. unconditionally takes its next instruction from the specified location and proceed from there (the control jumps), or conditionally jumps or does not jump to the specified location.

The addition or subtraction in the index registers is in modulo 10,000.

514 NOP .. - "No Operation" (0.30 + Ai)

This instruction causes the computer to take its next instruction in normal sequence.

N.B.1. NOP/ (515) clears the c(AC).

2. Numbers in the index and address parts are neglected.

710 HJM IJ A "Halt and Jump" (0.35 + Ai)

E → c(LC), and Halt.

E (= #IJA) replaces the c(LC) and the computation stops. When the START or SS button on the console is depressed, the computer takes an instruction from E and proceed from there.

714 JMP IJ A "Jump" (0.35 + Ai)

E → c(LC).

The control jumps to E (= #IJA) unconditionally.

750 JMI IJ A "Jump on Minus" (0.35 + Ai)

E → c(LC), if c(AC) ≦ -0, or normal sequence if c(AC) ≧ +0.

The control jumps to E (= #IJA) if the AC sign is minus, or normal sequence if plus.

752 JUN IJ A "Jump on UA No Zero" (0.35 + Ai)

E → c(LC), if c(UA) ≠ 0, or normal sequence if c(UA) = 0.

The control jumps to E (= #IJA) if c(UA) ≠ 0, or normal sequence if the 12-digit c(UA) = +0 or −0. The c(LA) are neglected.

754 JNZ IJ A "Jump on No Zero" (0.35 + Ai)

E → c(LC), if c(AC) ≠ 0, or normal sequence if c(AC) = 0.

The control jumps to E (= #IJA) if c(AC) ≠ 0, or normal sequence if the 23-digit c(AC) = +0 or −0.

756 JOV IJ A "Jump on Overflow" (0.35 + Ai)

E → c(LC), if c(UA)v ≠ 0, or normal sequence if c(UA)v = 0.

The control jumps to E (= #IJA) if the overflow position (v) of the UA contains a non-zero number, or normal sequence if contains zero.

N.B.1. This instruction is mainly used for a fixed point number. For a floating point number it is equivalent to "Jump on Exponent Plus".

758 JEO IJ A "Jump on Exponent Overflow" (0.35 + Ai)

E → c(LC), if c(UA)v ≠ 0 or 1, or normal sequence if c(UA) = 0 or 1.

The control jumps to E (= #IJA) if the characteristic of the c(AC) overflows or underflows (i.e. c(UA)v ≠ 0 or 1), or normal sequence unless the above.

712 JSW SJ A "Jump by Switch" (0.35 + Ai)

E → c(LC), if S is on, or normal sequence if off.

The control jumps to E (= #JA) if the JSW switch of No. S on the console has been set on, or normal sequence if set off. There are five switches, each of which corresponds to S = 1, ..., 5 respectively.

N.B.1. Any JSW switches can be set at a time.

2. If S = 6, ..., 9, 0, the operation equals to NOP(514).

850 JXL HJ A "Jump with Index Lowed" (0.35 + Ai)

If c(H) ≠ 0, E → c(LC), and c(H) - 1 → c(H).

Or if c(H) = 0, normal sequence, and 9999 → c(H).

If the c(H), contents of the index register specified ty H, are not zeros, the control jumps to E (= #JA). Or if the c(H) = 0, the control is in normal sequence. In either cases and at the end of the operation, the c(H) is decreased by one (modulo 10,000).

N.B.1. The H can be 1, 2 or 3 which specifys the IR1, 2 or 3.
If H = 4, ..., 9, 0, the operation equals to NOP (514).

2. The J can be as ordinarily 1,2,3 or 4. The modification takes place at the beginning of the operation. So if H = J, the value of the c(H) before it is changed is used for the #JA.

852 JXR HJ A "Jump with Index Raised" (0.35 + Ai)

If c(H) ≠ 0, E → c(LC), and c(H) +1 → c(H).

Or if c(H) = 0, normal sequence, and 1 → c(H).

Instead of decreasing the c(H) in case of JXL, this instruction JXR increases the c(H) by one in modulo 10,000. This is the only difference. So refer to JXL.

854 JXU HJ A "Jump on Index Unequal" (0.35 + Ai)

If c(H) ≠ c(IR1), E → c(LC), and c(H) + 1 → c(H).

Or if c(H) = c(IR1), normal sequnce, and c(H) + 1 → c(H).

If the c(H) ≠ c(IR1), the control jumps to E (= #JA). Or if the c(H) = c(IR1), the control is in normal sequence. In either cases and at the end of the operation, the c(H) is increased by one (modulo 10,000).

N.B.1. The H can be 1,2 or 3 which specifys the IR1, 2 or 3. If H = 1, the control is always in normal sequence. If H = 4, ..., 9, 0, the operation equals to NOP (514).

2. The J can be as ordinarily 1, 2, 3 or 4. The modification takes place at the beginning of the operation. So if H = J, the value of the c(IR1) before it is increased is used for the # JA.

856 JSX HJ A "Jump and Set Index from LC" (0.35 + Ai)

c(LC) → c(H), and E → c(LC).

The c(LC), i.e. the address number where this instruction is stored, replace the c(H), then the control jumps to E (= #JA).

N.B.1. The H can be 1, 2 or 3 which specifys the IR1, 2 or 3. If H = 4, ..., 9, 0, the operation equals to NOP (514).

2. The J can be ordinarily 1, 2, 3 or 4. The modification takes place at the beginning of the operation. So if H = J, the value of the c(J) before it is replaced is used for the #JA.

324 CMP IJ A "Compare" (0.55 + Ai + Aa)

If $c(MD) \begin{matrix} > \\ = \\ < \end{matrix} c(E)$, $c(LC) \begin{matrix} +1 \\ +2 \\ +3 \end{matrix} \rightarrow c(LC)$.

The c(MD) are compared with the c(E). If the c(MD) is greater than the c(E), the control is in normal sequence, or if equal, skips the next instruction, or if smaller, skips the next two instructions and proceed from there.

N.B.1. This instruction can be used for both fixed and normalized form floating-point numbers.

2. The contents of any registers, including the c(E), c(MD) and c(AC) are unchanged.

3. For comparision. the fixed point algebraic subtraction, c(MD) - c(E) is made (at the MQ). The sign of the difference decide the type of skips of the control.

4. Special care should be taken for the inclusion of a number in the overflow bit of the MD as well as of the register of loc. E in the fixed-point subtraction. So $|c(MD)|$ or $|c(E)| < 2$. This makes possible the comparision of two normalized form floating numbers by the fixed point subtraction.

5. This instruction regards; +0 > -0.

11. Index Operations

The following instructions are described; SEX, RAX, LWX, LXA, STX.

The instructions JXR, JXL, JXU and JSX are described in the Control Operations.

All the above instructions specify one of the three index registers by a number H = 1, 2 or 3 in the Xx position of a instruction which corresponds to the IR1, 2 or 3 respectively.

If the above H = 4, ..., 9, 0, all the above instructions equal to NOP (514) in their operations.

The address modification by the contents of the IR1, 2, 3 or LC is as ordinarily possible, and the address modification is made at the very beginning of the operation, so the contents on index registers are not yet changed by the instruction itself at the time of the modification.

An index register has four digits without sign, so all additions or subtractions are in modulo 10,000.

830 SEX HJ n "Set Index" (0.35 + Ai)

#Jn → c(H).

The #Jn (≡ c(~~J~~) + n in modulo 10,000) replaces the c(H).

N.B.1. If J = H, the c(H) are increased by n.

832 RAX HJ n "Raise Index" (0.35 + Ai)

c(H) + #Jn → c(H).

The c(H) are increased by #Jn (≡ c(J) + n; in modulo 10,000).

N.B.1. If J = H, 2 * c(H) + n replaces the c(H).

2. The following example increases the c(H) by c(#IJA)ad.

```
PSX IJ A
RAX HO 0      c(H) + c(#IJA)ad → c(H).
```

834 LWX HJ n "Lower Index" (0.35 + Ai)

c(H) - #Jn → c(H).

The c(H) are decreased by #Jn (≡ c(J) + n; in modulo 10,000).

N.B.1. If J = H, 10,000 - n replaces the c(H).

2. The following example decreases the c(H) by c(#IJA)ad.

```
PSX IJ A
LWX HO 0        c(H) - c(#IJA)ad  → c(H).
```

820 LXA HJ A "Load Index" (0.35 + Ai + Aa)

c(E)ad → c(H).

The address part of the c(E),(E = #JA), replaces the c(H) and other parts are unchnged.

822 STX HJ A "Store Index" (0.35 + Ai + Aa)

c(H) →c(E)ad.

The c(H) replaces the address part of the c(E), and other parts are unchanged.

12. Logical Operations

See chapter 3.

13. Special Operations

See chapter 3.

14. Input-Output Operations and Tape Control Codes

The following instructions are described; SEL, RIN with TCC, WRT, FWR, WSP. They select necessary input/output components, read instructions or data, and write the c(UA) or characters.

Characters handled by the computer are listed in the Table I KDC-I I/O Characters, (Conversion Table of Characters are also listed). Those codes on paper tape are listed in the Table II Paper Tape Character Coding.

The read operation is greatly facilitated by the help of special Input/Output Characters. These characters are named "Tape Control Codes" or abbreviated as TCC, though a name "Control Tape Codes" describes the situation better.

630 SEL IJ n "Select Component" (0.35 + Ai)

The input or output components are selceted by this instruction, the type of which is specified by #IJn as follows;

#IJn =	1111	Keyboard (KB)
	1112	Mechanical Tape Reader (MTR)
	1210	Photoelectric Tape Reader No.1 (PTR 1)
	1220	Photoelectric Tape Reader No.2 (PTR 2)
	0111	Printer (PR)
	0112	Punch (PU)
	0113	Both Printer and Punch simultaneously

The selection of an input component is continued only until a next selection of a different input component. The selection of an output component or components is also continued only until a next selection of a different output component or components.

The above components can be also selected manually by depressing the corresponding buttons on the computer console.

N.B.1. The execution of this instruction is suspended during the I/O operations by former instructions.

6

632 RIN IJ n "Read-in" (PTR; 200 characters/sec)

This instruction reads Input/Output characters in either a numerical mode (denoted as nu-mode) or the alphanumerical mode (denoted as alpha-mode) from the pre-selected input component, shifts the c(UA) to the left, converts codes of characters into a number as is specified in the Table of Input/Output (I/O) characters, and places the converted number into a vacated positions of the UA.

A number of characters read is specified by a number of the less significant two digits of #IJn, denoted as DU(#IJn).

The mode is specified by a number in the most significant fourth digit of #IJn. If it contains 0, the operation is in nu-mode, or if one,in alpha-mode.

However,the read operation is greatly influenced when TCC are read.

N.B.1. The c(LA) are unchanged.

2. DU(#IJn) $\leqq$ 23, or maximum of 23 characters can be read. If DU(#IJn) $\leqq$ 24, it is interpreted case be case into a number still less than 24. If DU(#IJn) = 0, equivalent to NOP.

3. The third digit of #IJn is neglected in the operation.

4. In the nu-mode a character is converted into a one-digit number while in the alpha-mode into a two-digit number, which are specified in the Table of I/O Characters. So ordinarily 11 or less characters are read in the nu-mode, while five or less characters are read in the alpha-mode.

5. Particular codes NE, ER, SP, CR or LF in the Table of I/O Characters are ignored by the computer.

6. Whenever the TCC " or ↺ is read, the read operation is ended. But the detail is described later.

7. In ordinary input routines, "RIN/ 00 14" or "Clear Read-in 14 digits" is used.

8. The execution of RIN is briefly as follows; a character is read and temporarily stored in the Input Register. If it is not a TCC or an ignored code, then the c(UA) are shifted to the left (same as SLS), by one place (nu-mode) or two (alpha-mode), and the codes of characters in the Input Register are converted into a specified number, and this number is placed in the vacated position of the UA, i.e. the 1 position (nu-mode) or 2 and 1 position (alpha-mode). The next character is read in the same way, and the process is repeated until

a number of characters, except the ignored codes (TCC are counted), reaches the specified number #IJn.
The case where TCC are read is described below.

TCC (Tape Control Codes); +, -, ", w, α, β, γ, δ.

Eight TCC exist, they facilitate the reading of instructions, fixed or floating point numbers to the UA, modify the address part of instructions, and causes the control to jump only in case of the nu-mode. In alpha-mode, they perform no special functions and are only ordinary characters.

(1) " End Mark

Whenever this code is read, the read operation is ended. So this code is used to show the end of words. For example RIN/ 00 14 can read 14 digits or less.

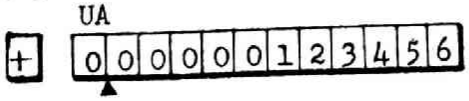
(2) +, - Sign Codes

These codes facilitate the reading of fixed point number and floating point numbers. On paper tape, a fixed point number should be expressed in the form; ± fraction " , while a floating point number should be expressed in the form; ± mantissa ± exponent " . Both can be read by the same RIN/ 00 14 instruction. Accordingly a sign code, if read for the first time by RIN or RIN/, is placed in the AC sign. If the second sign code is read, the computer regards it is reading a floating point number, and numbers before this second sign code are treated as a mantissa, and the successive characters till the end of the operation are read and treated as an exponent. Moreover a normalization is made in the UA not in the whole AC.

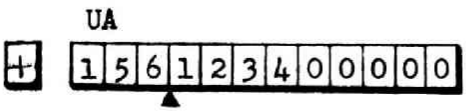
N.B.1. A fixed point number should be 12 digits or less, otherwise a real overflow occurs. A mantissa and an exponent of a floating point number should be 9 digits or less and 2 digits or less respectively, otherwise an erraneous result replaces the c(UA) in rather a case-by-case fashion. An exponent of -100 should be expressed as -00.

2. +0+0", +0-0", +0", 0", ", all of them result in the same fixed or floating zero in the UA if read by RIN/ 00 14.

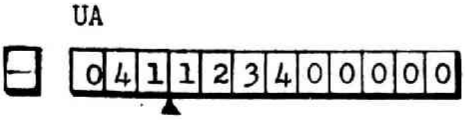
e.g.1. +123456" on paper tape is read by RIN/ 00 14 as follows;



e.g.2. +1234456"



e.g.3. -0001234-56"

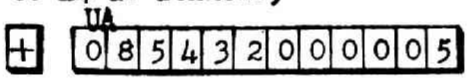


(3) w Address Read-in Code

This code facilitate the reading of instructions. On paper tape, an instruction should be expressed in the form; 3-digit function part, 2-digit index part, 1-digit break-point part, TCC w, 4-digit or less digit address part, and TCC ".

The execution is briefly as follows; if TCC w is detected, the cyclic shift to the left by 5 places of the 12-digit UA takes place, and successive characters are temporarily read into the MQ until TCC " or δ is detected. If detected c(MQ)ad is added into the c(UA)ad in modulo 10,000.

e.g.1. An instruction JXU 320 5 is punched on tape as 854 320 w 5 " and is read by RIN/ 00 14 as follows;

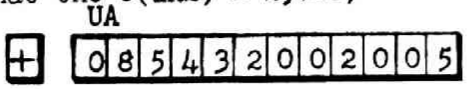


(4) α, β, γ Modified Address Read-in Codes

These codes also facilitate the reading of instructions like TCC w. These codes not only perform the whole function of TCC w, but also modify the address part of a just read-in instruction at the UA by the contents of the IR1(α), IR2(β) and IR3(γ).

To descriminate the above three conditions, TCC α sets on a-flip-flop, β sets on b-flip-flop, and γ set on both a and b flip-flops at the time these codes are read, and these flip-flops are set off at the end of the operation.

e.g.1. 854 320 β 5 " on paper tape is read by RIN/ 00 14 as follows; (provided that the c(IR2) = 2,000)



(5) δ LC-Setting Code

This instruction causes the control of the computer to jump to the location of a just read-in address part number of the c(UA) and the read-in operation is ended.

e.g. 3850 δ on paper tape is read by RIN/ 00 14, then 3850 replaces the c(UA)ad and the control jumps to loc. 3850.

e.g. δ on paper tape is read by RIN/ 00 14, since the c(AC) is cleared, the control jumps to loc. 0.

N.B.1. This instruction is equivalent to "jump to loc c(UA)ad and TCC ". So if the c(UA) are shifted cyclic by 5 places and characters are temporarily read into the MQ by TCC w, α, β, or γ, the c(UA)ad, not the c(MQ)ad, is the location to which the control jumps.

634 WRT IJ n "Write" (0.35 + Ai; 8 characters/sec)

This instruction causes the computer to store the c(UA) temporarily in the buffer register (BR), convert them into the Input/Output (I/O) Characters in either a numerical mode (nu-mode) or an alphanumerical (alpha-mode), and print and / or punch these characters via pre-selected output components.

A number of characters written is specified by a number of the less significant two digits of #IJn, denoted as DU(#IJn).

The mode is specified by a number in the most significant fourth digit of #IJn. If it contains 0, the operation is in nu-mode, or if one, in the alpha-mode.

N.B.1. The c(UA) and c(LA) are unchanged.

2. In the nu-mode, a number in the m position of the UA is written as a first character, 10 position next, and so on. The DU(#IJn) should be 11 or less.

3. In the alpha-mode, a two-digit number in 10 and 9 positions of the UA are converted into a character specified by the Table of I/O Characters and is written on papers, 8 and 7 positions next, and so on. DU(#IJn) is a number of characters and not a number of digits in the UA. The DU(#IJn) should be 5 or less.

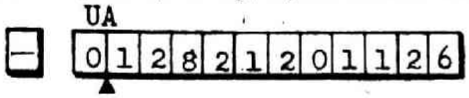
4. The c(UA)v is neglected in the nu-modes, while c(UA)v,m are neglected in the alpha-modes.

5. The AC sign is not written out. It should be written by WSP after judging the AC sign.

6. The operation is concurrent, i.e. after the c(UA) are stored in the buffer register, the write operation begins, and at the same time the computer proceeds from the next instruction. If another write instruction comes before the operation of this write instruction, the execution of the next one is suspended until the former one finishes.

7. The operation time without suspension time and actual write time is thus 0. ms, and the actual write time is 120 * #IJn ms .

8. If DU(#IJn) $\geq$ 12 or 6 in the nu or alpha-mode respectively, the situation is rather a case-by-case and it is not described here. If DU(#IJn) = 0, equivalent to NOP. fashion

e.g. 

→ WRT 00 0006 → 128212
→ WRT 00 1005 → KDC-I

638 FWR .. - "Floating Write" (0.35 + Ai; 8 characters/sec)

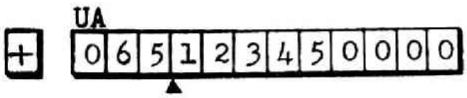
The c(UA) are treated as a floating point number, written out in the form ± mantissa ± exponent via pre-selected output components.

N.B.1. The (AC) are unchanged. The normalization is not made.

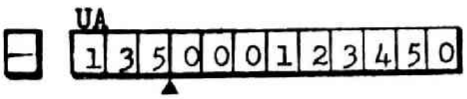
2. If a characteristic is zero or an exponent is -100, the exponent written out is -00.

3. If the c(UA) =+0, its result is; +000000000-00

4. Numbers in the index part and the address part of the instruction are neglected in the operation.

e.g.1. 

+123450000-35

e.g.2. 

-000123450+35

636 WSP IJ m "Write Special" (0.35 + Ai; 8 characters/sec)

This instruction writes a character, or a multiple of the same character, of a type specified by the fourth and third digits of # IJm, denoted as KH(#IJm), and a number of multiplitude is specified by the less significant first and second digits, i.e. DU(#IJm). Characters which correspond to KH(#IJm) are listed in the Table of Input/Output Cahracters.

N.B.1. DU(#IJm) should be 62 or less, if zero, equivalent to NOP.

N.B.2. If DU(#IJm) > 62, or if KH(#IJm) is not a number listed in the Table of I/O Characters, the situation is rather a case-by-case fashion and it is not described here.

e.g. WSP 00 2801
WSP 00 2101
WSP 00 2001
WSP 00 1101
WSP 00 2601
The following characters are written;- KDC-I

15. Magnetic Tape Operations

See chapter 4.